Robust Mask-layout and Process Synthesis in Micro-Electro-Mechanical-Systems (MEMS) Using Genetic Algorithms

Thesis by
Lin Ma

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

2001

(Defended May 11, 2001)

Acknowledgements

I would like to express my gratitude to my advisor, Professor Erik Antonsson, for his guidence and encouragement. His patience and optimism make my work and study at Caltech a pleasant experience. Certainly without his advice this work will be impossible.

Also I would like to thank my committee members, Professor Joel Burdick, Professor David Goodwin, and Professor Kennith Pickar, for the time they spend on guiding me through and their inspiring ideas.

I owe my special thanks to Qi, for her love and support. I also thank my parents, who always take pride in me.

Robust Mask-layout and Process Synthesis in Micro-Electro-Mechanical-Systems (MEMS) Using Genetic Algorithms

by

Lin Ma

In Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

Abstract

This thesis reports a Genetic Algorithm approach for the mask-layout and process flow synthesis problem. For a given desired target shape, an optimal mask-layout and process flow can be automatically generated using the Genetic Algorithm synthesis approach. The Genetic Algorithm manipulates and evolves a population of candidate solutions (mask-layouts and process parameters) by utilizing a process simulation tool to evaluate the performance of the candidate solutions. For the mask-layout and process flow synthesis problem, encoding schemes, selection schemes, and genetic operations have been developed to effectively explore the solution space and control the evolution and convergence of the solutions.

The synthesis approach is tested for mask-layout and process synthesis for bulk wet etching. By integrating a bulk wet etching simulation tool into the Genetic Algorithm iterations, the algorithm can automatically generate proper mask-layout and process flow which can fabricate 3-D geometry close to the desired 3-D target shape. For structures with convex corners, complex compensation structures can be synthesized by the algorithm. More importantly, the process flow can also be synthesized. For multi-step wet etching processes, proper etchant sequence and etch times for each etch step can be synthesized automatically by the algorithm. When the choice of different process flows exists, the enlarged solution space makes the design problem more challenging. The ability to synthesize process flows makes the automatic design method more complete and more valuable.

The algorithm is further extended to achieve robust design. Since fabrication variations and modeling inaccuracy always exist, the synthesized solutions without considering these variations may not generate satisfactory results in actual fabrication. Robust design methods are developed to synthesize robust mask-layouts and process flows in "noisy" environment. Since the synthesis procedure considers the effect of variations in the fabrication procedures, the final synthesized solution will have high robustness to the variations, and will generate satisfactory results under a variety of fabrication conditions. The robust design approaches are implemented and tested for robust mask-layout design for mask misalignment and etch rate variations. Mask-layouts robust to mask misalignment noise and etch rate variations during the fabrication can be synthesized. The synthesized mask-layouts generally improve the yield significantly by exhibiting consistent performance under a variety of fabrication conditions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the variety and complexity of micro-electro-mechanical-systems (MEMS) expand, the need for design tools and automation strategies to create robust, cost-effective, and manufacturable micro-machined devices and systems becomes critical. Such tools will allow non-specialists to design sophisticated MEMS devices, and will enhance the efficiency of expert designers. While there has been diverse work in the general area of computer aided design (CAD) for MEMS, many of these tools apply to process simulation or analysis of the behavioral and mechanical properties of devices. While process simulators and device analysis tools are excellent tools for verification and refinement, and they can reduce cost and time of device development, the task of developing the initial process (including mask-layouts), and appropriate changes to the process if the simulation results are unsatisfactory, remains on the designer. The design process itself, for mask-layouts and fabrication process flows, is still a job for experts, and iterations of trials and errors are needed to achieve a robust and effective procedure.

At present, a MEMS designer conceives of a function, then informally creates a mask-layout and a fabrication process flow that the designer believes will manufacture a shape that will exhibit the desired function. This process is based largely on the designer's intuition and experience, and is difficult for complex geometries or for novice engineers. A prototype device is created from the suggested mask and fabrication process, and its actual function is tested. If the performance of the device is unsatisfactory, changes must be made to the mask-layout and fabrication process. This procedure results in many iterations and many prototypes. Process simulation tools and function modeling tools are developed to simulate the fabrication procedure and test the device performance on computers. Although these tools can speed up the device development and provide verification for the design, they do not provide much help for the mask-layout and process synthesis task, which still relies heavily upon the designer's intuitive understanding of the fabrication process. The designer can only count on experience, and the simulation results, to design and modify the mask-layouts and fabrication processes. Additionally, the wide variety of micromachining processes in use today makes this intuitive work even more difficult, and typically force designers to limit themselves to familiar processes. In some cases certain etchants or geometries are avoided because they require too many iterations to get the right mask-layouts and process flows.

The desired approach for designing MEMS devices is the reverse of the process just described: The designer conceives of a MEMS function, then through an automated process determines a shape (three-dimensional geometry) that will exhibit the desired function. Another automated process is then used to determine a mask-layout and a set of fabrication instructions that will create the desired shape. Thus the present process is: mask-to-shape-to-function, the desired process is the reverse: function-to-shape-to-mask.

This thesis addresses issues in formalizing and automating the synthesis (design) of MEMS, and discusses the development of algorithms that automate the robust mask-layout and process design for bulk wet etching.

1.1 MEMS Design Approaches

1.1.1 Overview

The development for MEMS devices benefits from the use of computer modeling and simulation. Most of the design tools developed for conventional mechanical engineering can be utilized to assist in the mechanical design aspects of MEMS devices, and various process simulation tools and device analysis programs have been developed to provide help in the design process. However, micromachining process synthesis and design automation are still largely lacking. Therefore, a major portion of the design cycle may still include time-consuming experimental determinations of design space and process improvement to ensure manufacturing accuracy and robustness. New design tools and automation strategies are needed to provide robust, cost-effective, fast turn-around, and manufacturable MEMS products [113]. The ability to synthesize mask-layout and process flows and device geometries from function specification will represent the ultimate design automation of micromechanical systems.

For MEMS device development, there exist three different approaches. They are described below, and the advantages of the synthesis and optimization approach are discussed.

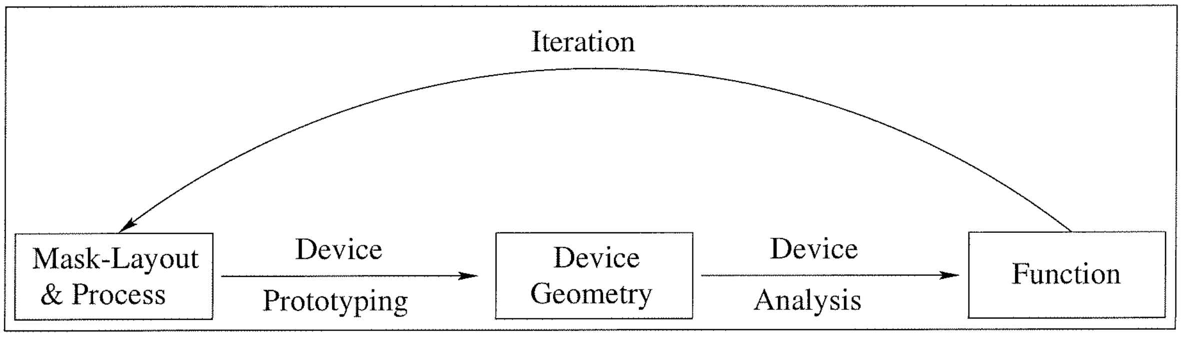


Figure 1.1: MEMS design: empirical approach

1.1.2 Empirical Approach

A typical development path for a MEMS device with an empirical approach without the aid of computer simulation is illustrated in Figure 1.1. Designers start with the mask-layout and fabrication process flow. A prototype is fabricated and tested, and the performance is compared with the target or desired function. The first prototype's performance is in general unsatisfactory, and iterations of device prototyping and analysis process are usually involved to move towards an optimum design solution. The entire iteration procedure can be broken into two sub-procedures each of which takes the form of iterations: the device analysis step, and device prototyping step. Iterations of device analysis steps are taken until a satisfactory device geometry is found which meets the performance function requirement, and iterations of device prototyping are taken with the goal of searching for a mask-layout and fabrication process flow that will produce the desired device shape. Device prototyping iterations start with the selection of a fabrication process and the settings of process inputs, which typically include material, mask-layout and process parameters such as temperature and process duration time. Then the actual fabrication proceeds to produce the device prototype. The prototype is then compared with the desired device shape, and if the shape comparison result doesn't meet the required tolerance, readjustment of mask-layout and process parameters need to be made before starting next iteration.

This procedure depends strongly on human experience in the involved MEMS design and fabrication processes and such experience is usually obtained through many repeated trials and errors. Because of the many iterations of device prototyping and analysis, the production yield is low.

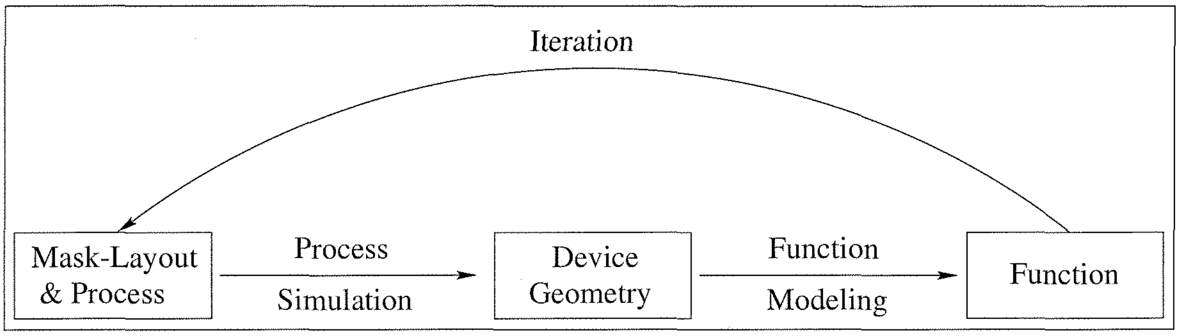


Figure 1.2: MEMS design: simulation and modeling approach

1.1.3 Simulation and Modeling Approach

Several steps of MEMS design process can be automated which makes the iterations faster and cheaper. Figure 1.2 illustrates how computer simulation can be used to reduce costly fabrication and testing iterations. There are two categories of simulation: process simulation (simulating fabrication processes), and function modeling (analyzing device performance).

Process simulation tools can be used to automate the fabrication step and, when combined with function modeling tools (finite element analysis, *etc.*), further automate the device analysis step. Combining these two, the device prototyping experience can be learned through the simulation on computers instead of actual device production with many trials and errors, and the device performance assessment can be performed without actually testing the devices.

Process Simulation

A process simulation tool takes mask-layout and fabrication process settings as input and generates an accurate three-dimensional representation of the resulting device geometry to take the place of the actual fabricated device geometry.

The majority of bulk-micromachining processes involve isotropic or anisotropic etching of single-crystalline silicon, gallium arsenide, or amorphous glass. For anisotropic etching, the etch rates at different crystalline directions are different, and the transition from the mask-layout to the etched shape is generally geometrically complex. An accurate etch rate model is essential to simulate the fabrication process. A good bulk etching simulation tool can be useful to predict the device geometry from the mask-layout. For example, the ability to model the formation of silicon convex corners with KOH etching will help in designing corner compensation structures.

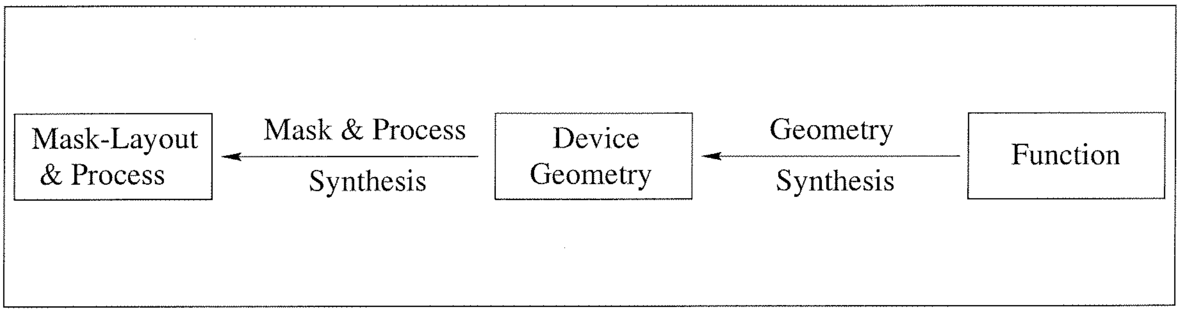


Figure 1.3: MEMS design: synthesis and optimization approach

Surface micromachined devices, on the other hand, are built from thin film materials such as polycrystalline or amorphous silicon, silicon nitride, silicon dioxide, and various metal thin films. An ideal process simulator will be able to simulate the geometric effects of sequential IC process stages, including patterning of photoresists, oxidation, diffusion, deposition, etching, *etc.*, to produce geometric representations of the structure after each process stage.

Function Modeling

The purpose of the device design is to develop a device which has some desired functional performance. Microfabricated sensors function by converting one or several physical or chemical conditions into electrical or optical signals, while most actuators perform the inverse functions. The modeling of MEMS devices generally involve electrostatic, electromagnetic, ferroelectric, and piezoelectric phenomena, to name a few. These characteristics are often coupled with static or dynamic mechanical deformation of the microstructures. Therefore, a self-consistent modeling tool is needed to represent the transfer behaviors of these devices. For most micromechanical structures, the conventional CAD analysis tools, *e.g.*, the finite element method, can be used to analyze the mechanical behavior.

1.1.4 Synthesis and Optimization Approach

Even though both fabrication and device analysis steps can be automated, without the automation of the search through the design space (composed of mask-layouts and fabrication process flows), designers still can only count on experience to drive the entire iterative device prototyping and analysis process. Development of automatic geometry synthesis and mask-layout and process synthesis will lead to the ultimate design automation of micromechanical systems, as illustrated in Figure 1.3.

These synthesis and optimization programs will automatically generate the device geometry and fabrication process for such device given a desired device functionality. Recently, various process synthesis tools have been developed to automatically synthesize the fabrication processes for a given device design [35, 47, 52, 82, 91]. For device geometry synthesis, although currently there is no existent software tools directly targeting it, desires have been raised to develop future geometry synthesis tools which will automatically synthesize the optimum design solution for the targeted functional features. By utilizing process synthesis and geometry synthesis tools, the burden of MEMS fabrication considerations is reduced so that designers can fully focus on the device functional design.

1.2 Current MEMS CAD Research

As illustrated above, as the field of microelectromechanical systems develops there is an increasing need for CAD systems to assist designers of devices and systems in a variety of tasks. Current research on MEMS CAD is investigating a number of tools for MEMS design. Research work on process synthesis and design automation will be described in the next section, and the following discusses CAD tools for process simulation and device analysis.

1.2.1 Process Simulation Tools

As microelectromechanical systems become more complex, designers will find it useful to derive models of the geometry of their device structures directly from the process description and planar mask patterns. With process simulation tools, MEMS designers can verify process descriptions and masks before beginning fabrication, and ensure a robust design by examining performance sensitivities to process and mask variations.

The development of process simulation tools is directly influenced by MEMS fabrication technologies. Currently, there are mainly two types of MEMS fabrication techniques: surface micromachining, and bulk micromachining.

Surface micromachining

For surface micromachining, most of the techniques can be borrowed from IC processes which have been well-developed through the past decades. Therefore, most process simulation tools of surface micromachining are obtained through slight modification of existing VLSI simulators. These simulators cover standard semiconductor processes such as oxidation, diffusion, thin-film deposition, and plasma etching.

For example, a simulator called OYSTER was developed by Koppelman [64]. OYSTER simulates the geometric effects of sequential IC process stages, in order to produce three-dimensional polyhedral representations of all material structures in a design cell after each process stage. The polyhedral models may be used with various analytic procedures as sources of geometric data for finite-element calculations, or they may be subjected to interference calculations, or inspected to detect structural anomalies. Another example is SAMPLE for simulation of projection lithography deposition, and etching [85, 86].

Bulk micromachining

Bulk micromachining is used to fabricate three-dimensional mechanical structures (especially high aspect-ratio structures) on the silicon substrate with the use of wet (chemical) or dry (plasma) etching technique combined with masking films or etch-resistant layers.

Most of the chemicals used today are anisotropic etchants which attack different crystallographic orientation of the material at different rates and provide control over the intended shape of the wafer. Wet isotropic etching and plasma etching are also used to provide geometries that are not dependent on crystallographic orientation.

All bulk micromachining simulation programs predict the three-dimensional geometry change of the underlying etching structure for different etching time steps based on the input of a mask-layout and etch rate data. Early etch modeling work includes the Wulff-Jaccodine method of traveling planes [42, 59]. A number of computer implementations have been developed [17, 18, 28, 29, 107, 108, 115]. Another important etching simulation tool is the Slowness method by Foote and Sequin [106]. The models of the Wulff-Jaccodine method and the Slowness method are purely based on geometric analysis, while Hubbard and Antonsson [56], Than and Buttgenbach [114], and Zhu and Liu [125, 126] developed cellular automata (CA) algorithms which base the rate of removal of each cubic cell of material on the number of nearest neighbor cells remaining. The geometric analysis is fast and accurate but is limited to basic shapes; the cellular automata approach is slow but is able to simulate the etching of shapes of any complexity. SEGS by Hubbard and Antonsson [55] uses an edge segmentation approach to trade-off the advantages of the geometric analysis and cellular automata.

1.2.2 Device Analysis Tools

Fortunately, the conventional mechanical CAD tools can be borrowed for MEMS device analysis. The mechanical behavior analysis usually starts with a simulated three-dimensional solid model along with specified boundary conditions; the discretization process is followed to mesh the solid model. A PDE solver is utilized to generate evaluation results for performance properties such as stress and deformation distributions. The whole process is rather computationally intensive, and a major effort is being devoted to developing fast and accurate algorithms to improve the efficiency of the process. The mixed mode simulation and analysis are also developed where electrical and mechanical models are combined. Senturia *et al.* studied such mixed mode simulation such as capacitance calculations for a mechanically deformed diaphragm [104].

1.2.3 MEMS CAD Packages

Much work has been done on the broader picture of MEMS design concerning CAD architecture and the interface of process simulators with function modelers [16, 64, 74, 93, 105, 118].

Various MEMS CAD programs have been developed to integrate process simulation tools and device analysis tools. The examples are MEMCAD by MIT [105], IntelliCAD by IntelliSense [75] and CAEMEMS by the University of Michigan [16, 25, 124]. Commercial MEMS CAD packages have been developed by companies such as Coventer (originally Microcosm), IntelliSense, MEMSCAP and CFDRC. These programs typically also integrate a geometric solid modeler at front for users to construct 3D structures and maintain material and process databases for users to specify materials and fabrication processes. With the use of such multi-functional tools, designers can carry out "what if" experiments during the device prototyping and analysis process more effectively. However, even under these integrated design environments, designers still need to undertake many trials in order to seek the optimum design solution and the appropriate process settings. Therefore, the development of synthesis tools to achieve a higher level of design automation becomes rather important.

1.3 Synthesis and Design

As demonstrated in last sections, the iterative nature of the simulation-based design methodology represents an increasing burden as process complexity grows. There is a clear need for automated design tools to develop fabrication process specifications from desired device geometry.

The design of VLSI systems has become highly formalized and automated since Mead and Conway's early work in this area [57, 78]. Prior to their work, VLSI design was the exclusive domain of highly trained and experienced specialists. Mead and Conway's synthesis approach enabled VLSI designers to concentrate on the desired function of the chip, rather than the details of its physical implementation. Other engineering domains (including MEMS design) have not had the benefit of the same level of formalism and automation in design, and engineering design in these areas remains the province of highly trained and experienced specialists. The great contribution of VLSI CAD synthesis tools to the success of VLSI technology provides a realistic image of the potential impact that MEMS synthesis tools can give to the future development of MEMS technology.

As far as surface micromachining is concerned, the structure of process flow is very similar to that of VLSI. Such similarity creates opportunities for the MEMS structured design tools to borrow much of the VLSI framework which includes the separated abstraction levels and the hierarchical system modeling based on primitive functional elements. The recent development of schematic synthesis design tools for surface micro-machining by Fedder *et al.* has demonstrated such benefits [35, 82]. Mastrangelo and his group at University of Michigan have developed MISTIC [47, 52], a process compiler for thin film micro devices. MISTIC employs a systematic method for the automatic generation of fabrication processes of thin film micro-machined devices. By using topological sorting techniques, all possible process sequences can be extracted from the layer order in terms of fundamental processing steps like deposition, lithography and etching, and an optimal sequence is determined by using a cost function based on a database of materials and processes. Such VLSI-like synthesis methods could be very helpful for the overall design of MEMS devices whose performance depends on their component layout and inter-connection between components. But for high aspect-ratio devices whose performance critically depends on their three-dimensional geometry, especially those fabricated by bulk micromachining, no synthesis tools exist that can automate the device design process.

In a typical bulk micromachining process, an amount of silicon is etched away to build a three-dimensional structure. The geometry of the three-dimensional structure is generally important to ensure device performance, and many of the most interesting, useful, and valuable mechanical devices intrinsically rely on three-dimensional behavior and three-dimensional shapes. But because of the anisotropicity of the etching, the transition from the mask-layout to the final shape is in general complex and non-intuitive. Some methods have been proposed to synthesize local mask-layout geometry for convex corners in device shapes by studying etch rate diagram and corner convexity [71], but generally speaking, for complex device shapes reversing a fabrication process simulation (so that a 2-D mask-layout might be produced) is impractical [54]. Furthermore, sometimes the fabrication process flow may also need to be synthesized. For example, complex three-dimensional shapes can be generated using fabrication procedures such as multi-step wet etching process [43, 63, 98, 109]. For a multi-step wet etching process design, the etchant sequence and etch duration for each etching step are design parameters that need to be determined, and to choose a right formula (process flow) is not an easy task because of the large number of different possibilities. A mask-layout and process synthesis methodology is badly needed to help the designer design proper mask-layout and fabrication process flow for a desired device shape.

1.4 Contributions of This Thesis

To attack the mask-layout and process synthesis problem, a Genetic Algorithm approach for mask-layout and process synthesis is adopted here that uses simulation of fabrication process in an iterative refinement loop. An optimum mask-layout and fabrication process flow will be automatically found through a stochastic search. A flowchart of this approach is shown in Figure 1.4. In this approach, the user specifies a desired three-dimensional shape. Initial candidate mask-layouts and fabrication process flows are randomly generated, and the fabrication is simulated through a process simulator. The simulated shape is then compared with the desired shape to decide if it's close enough, and if it's not, genetic operations are applied to refine the masks and process parameters, and the procedure is repeated. The iteration stops when the simulated shape is close enough to the desired shape. For the mask-layout and process flow synthesis problem, encoding schemes, selection schemes, and genetic operations have been developed to effectively explore the solution space and control the evolution and convergence of the solutions.

One of the primary benefits of this approach is that any fabrication process can be utilized with this synthesis method, as long as an efficient simulation program of the process exists. Thus synthesis can be performed on devices to be fabricated from bulk-wet etching as well as surface micromachining. This approach is tested for mask-layout and process synthesis for bulk wet etching. By integrating a bulk wet etching simulation tool into the Genetic Algorithm iterations, the algorithm can automatically generate proper mask-layout and process flow which when used can fabricate 3-D geometry close to the desired 3-D target shape. For structures with convex corners, complex compensation structures can be synthesized by the algorithm. More importantly, the process flow can

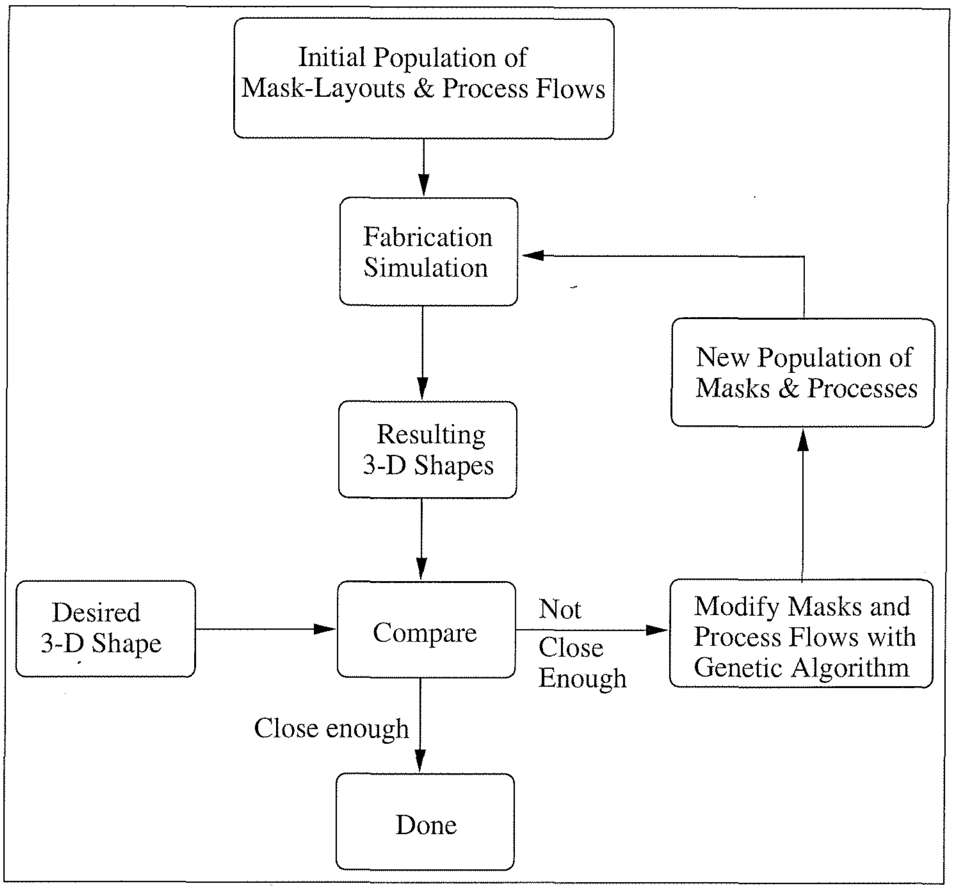

Initial Population of Mask-Layouts & Process Flows
Fabrication Simulation
New Population of Masks & Processes
Resulting 3-D Shapes
Desired 3-D Shape
Compare
Not Close Enough
Modify Masks and Process Flows with Genetic Algorithm
Close enough
Done

Figure 1.4: Iterative synthesis flowchart

also be synthesized. For multi-step wet etching processes, proper etchant sequence and etch times for each etch step can be synthesized automatically by the algorithm. When the choice of different process flows exists, the enlarged solution space makes the design problem more challenging. The ability to synthesize process flows significantly increases the value of the automatic design method.

The Genetic Algorithm synthesis approach is further extended to achieve robust design. Since fabrication variations and modeling inaccuracy always exist, the synthesized solutions may not generate satisfactory results without considering these variations. Robust design methods are developed to synthesize robust mask-layouts and process flows in "noisy" environment. By integrating expected variations (mask misalignment, etch rate variations, *etc.*) into the synthesis iteration, and setting robustness of the candidate solutions as one of the performance factors for candidate solution evaluation, the iterative stochastic optimization will produce mask-layouts and process flows that are least sensitive to these variations and can generate satisfactory results under a variety of fabrication conditions. The robust design approaches are implemented and tested for robust mask-layout design for mask misalignment and etch rate variations. Mask-layouts robust to mask misalignment noise and etch rate variations during the fabrication can be synthesized.

1.5 Overview of the Chapters

Chapter 2 gives an introduction to Genetic Algorithms, the optimization method used in the approach to the mask-layout and process synthesis. The general structure and various components of the Genetic Algorithm are described. Chapter 3 briefly describes the SEGS bulk wet etching simulation tool. In later chapters SEGS is integrated into the Genetic Algorithm iterations to implement an algorithm for mask-layout and process synthesis for bulk wet etching. Chapter 4 discusses issues in shape matching, and describes an algorithm to compare two 3-D shapes represented by polygonal contours. The shape matching algorithm is used in the synthesis to compare the shapes resulted from the simulation with a specified target shape. For the Genetic Algorithm approach to the mask-layout and process synthesis, the design and implementation details are provided in Chapter 5. Chapter 6 tests the Genetic Algorithm approach to the mask-layout and process synthesis for bulk wet etching by using SEGS as the simulator for wet etching steps. For specified target shapes, optimal mask-layouts and process settings are generated by the synthesis algorithm. Chapter 7 focuses on the robust design issues. Robust design methods are developed to extend the original GA synthesis method so that mask-layouts and processes robust to the fabrication variations and modeling inaccuracy can be synthesized. Tests of the algorithms are conducted for mask misalignment and etch rate variations. Major results are summarized in Chapter 8, which concludes the thesis.

Chapter 2

Genetic Algorithms

2.1 Introduction

Many optimization problems from science and engineering are complex in nature and difficult to solve by conventional optimization methods. Since 1960s, there has been an increasing interest in simulating the natural evolutionary process to solve optimization problems. Stochastic optimization techniques called *evolutionary algorithms* [8, 37] have been developed, which can often outperform conventional optimization methods when applied to difficult real-world problems. There are currently three main avenues of this research: Genetic Algorithms (GAs) [49, 53], Evolutionary Programming (EP) [38, 39], and Evolution Strategies (ES) [96, 100]. Among them, Genetic Algorithms are perhaps the most widely known type of evolutionary algorithms today and have been successfully applied to solve many science and engineering problems in various complex domains [49]. The well-known applications of GAs include scheduling and sequencing [23, 34, 97, 122], structure design [19, 48, 58, 60, 84], layout design [69, 70, 77, 99], and many others.

Genetic algorithms (GAs) are global stochastic optimization techniques that are based on the adaptive mechanics of natural selection evolution. They were invented by John Holland [53] in 1975, and subsequently have been made widely popular by David Goldberg [49]. GAs use two basic processes from evolution: *inheritance*, or the passing of features from one generation to the next, and competition, or *survival of the fittest*, which results in weeding out individuals with bad features from the population. In general, Genetic Algorithms are a non-problem specific technique which can be applied to virtually any optimization problem if its objective function measurement is available.

For many optimization problems, the performance landscape (fitness surface) has one or more of the following characteristics: (i) high nonlinearity, (ii) multiple performance optima, and (iii) non-analyticity (*i.e.*, cannot be written explicitly in a functional form). These characteristics preclude the use of gradient-like search techniques. Perhaps more troubling is that typical design spaces are so large that traditional search techniques, such as branch and bound, linear programming, *etc.*, are ineffective. GAs are particularly good at finding optima where the fitness surface is nonlinear, multi-modal, and dependent on several parameters simultaneously. Although in many cases a GA is unlikely to produce a globally optimal solution to a problem in a given amount of time, it can produce a solution that is close to the optimum in a time orders of magnitude less than some other algorithm such as exhaustive search will need. As a stochastic optimization method, Genetic Algorithms have many advantages over traditional techniques. GAs are adaptive search method; they maintain a population of solutions instead of single solution, and trapping into local optima can be avoided; the algorithm is implicitly parallel in the sense that a large number of different *schema* are evaluated simultaneously during a generation [53]; the computation can be easily parallelized for efficient execution; and finally, the algorithms impose virtually no mathematical requirements on the optimization problem: they can handle any kind of objective function and constraints. The following sections describe the general structure and key components of Genetic Algorithms, and the appendices provide more implementation details and the theoretical background. At the end of the chapter, as an example to show how to construct the various components of a Genetic Algorithm, a GA is constructed to search for the maximum of a function.

2.2 General Structure

A Genetic Algorithm maintains a population of solution candidates and works as an iteration loop. First, an initial population is generated randomly. Each individual in the population is an encoded form of a solution to the problem under consideration, called a chromosome which is usually a string of characters or symbols, *e.g.*, a string of 0's and 1's (a binary string). The chromosomes evolve through successive iterations, called generations. During each generation, the chromosomes are evaluated by a fitness evaluation function and selected according to the fitness values using a selection mechanism, *e.g.*, fitness-proportionate selection, so that fitter chromosomes have higher probabilities of being selected. New chromosomes, called offspring, are formed by either merging two selected chromosomes from the current generation using a crossover operator, or modifying a chromosome using a mutation operator. Crossover results in the exchange of genetic material between relatively fit members of the population, potentially leading to a better pool of solutions. Mutation randomly introduces new features into the population to ensure a more thorough exploration of the search space. A new generation is created by selecting chromosomes from the parents and the offspring. The population's average fitness will improve as this procedure continues, and the algorithm will converge to a best chromosome approaching the optimal or near-optimal solution. The general structure of Genetic Algorithms is shown in Figure 2.1 in flowchart form.

To use Genetic Algorithms, each of the following must be developed:

Encoding scheme. In GAs, a population of candidate solutions is maintained and manipulated by genetic operators. The solutions are encoded as chromosomes (usually strings of characters or symbols, *e.g.*, binary strings, real number strings, or symbol strings) to which genetic operators can be applied. An encoding scheme is needed to map candidate solutions into coded strings.

Initialization of population. The initialization is usually done randomly to sample the search space uniformly without bias. A well-initialized population can improve the algorithm's robustness and effectiveness in finding an optimal solution, while a poorly-initialized population may trap the algorithm in local optima and make it impossible to reach the global optimum.

Evaluation function. During the operation of Genetic Algorithms, all chromosomes are evaluated to see how fit they are as solutions to the problem. An evaluation function is required to assign a fitness value to each chromosome.

Genetic operators. In general the well-studied typical canonical genetic operators cannot be directly applied, since they only work with particular encoding schemes. Additionally, problem-specific information should be considered when designing efficient, meaningful operators. Genetic operators suitable to the problem and the encoding scheme must be developed.

2.3 GA Terminology

In this section, the terminology used in Genetic Algorithms is described. Since Genetic Algorithms simulate natural evolution, some terminology used in GA literature is borrowed from natural genetics.

Genetic Algorithms work on a *population*, or a collection of candidate solutions to the given problem. Solutions are encoded as *chromosomes* which are usually strings of characters and sym-

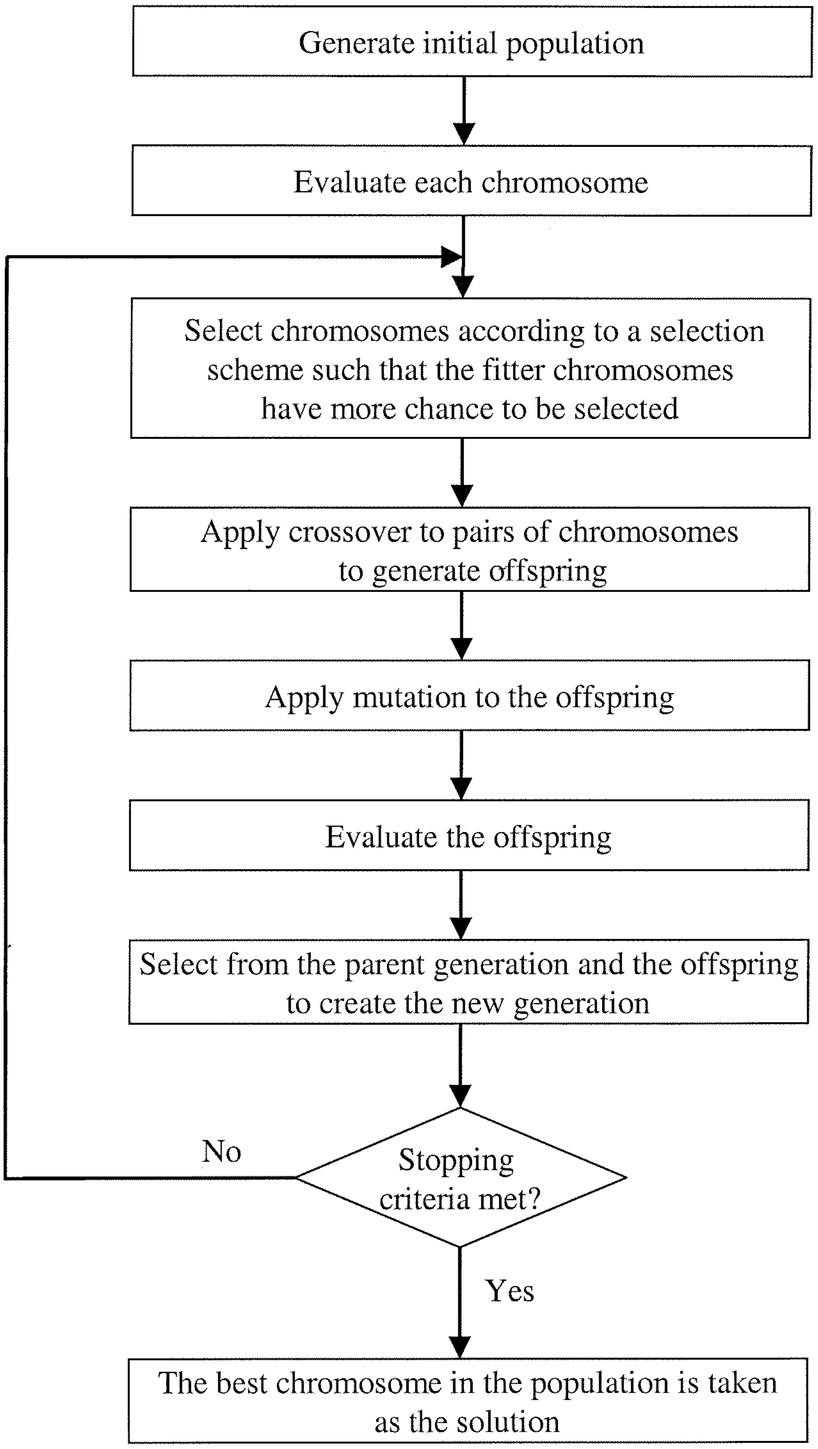

Generate initial population
Evaluate each chromosome
Select chromosomes according to a selection scheme such that the fitter chromosomes have more chance to be selected
Apply crossover to pairs of chromosomes to generate offspring
Apply mutation to the offspring
Evaluate the offspring
Select from the parent generation and the offspring to create the new generation
Stopping criteria met?
No
Yes
The best chromosome in the population is taken as the solution

Figure 2.1: The general structure of Genetic Algorithms

bols, and the individual characters or symbols in the strings are referred to as *genes.* Candidate solutions constitute the *phenotype space*, and chromosomes constitute the *genotype space.* An *encoding* scheme is the mapping function between the phenotype and genotype space. Each chromosome is evaluated by an *evaluation function* (or *fitness function*) to determine the *fitness* value. The evaluation function is usually user-defined, and problem-specific.

Individuals are selected from the population for reproduction, with the *selection* biased toward fitter individuals. Different *selection schemes* have been developed to ensure that survival of the fittest occurs. The selected individuals form pairs, called *parents*, and a genetic operator called *crossover* is applied to two parents to create two new individuals, called *offspring*. Crossover combines portions of two parents so that the offspring inherit a combination of the features of the parents. For each pair of parents, crossover is usually performed with a high probability P_C, which is called the *crossover probability.* With probability $1 - P_C$, crossover is not performed, and the offspring pair is the same as the parent pair. *Mutation* is another genetic operator applied to parents. Mutation randomly changes the chromosome in some way to introduce new features into a population and make the whole search space reachable. Mutation is usually applied with a low probability, P_M, called *mutation probability. Exploitation* and *Exploration* are the terms that refer to two different aspects during the search: exploiting the best solution and exploring the search space. Appendix A discusses exploitation and exploration in more details.

A new *generation* is created by selecting individuals from the parents and offspring. The *generation gap* is the fraction of individuals in the population that are replaced from one generation to the next. The generation gap is equal to 1 for *simple GAs* (also referred to as the *total replacement algorithm*) in which all individuals in the parent generation are replaced by the new offspring. In *steady-state GAs* [111], at each iteration only one pair of parents is selected to create offspring and then the two offspring replace the worst individual in the current population. Obviously in steady-state GAs, the generation gap is minimal, since only two offspring are produced in each generation.

Figure 2.2 illustrates the terminology. It should be noted that genetic operators (crossover and mutation) work on *genotype* space, while evaluation and selection work on *phenotype* space.

2.4 Encoding

Before a GA can be applied to a problem, a suitable *encoding* (or *representation*) for the problem must be developed. How to encode a solution of the problem into a chromosome is a key issue

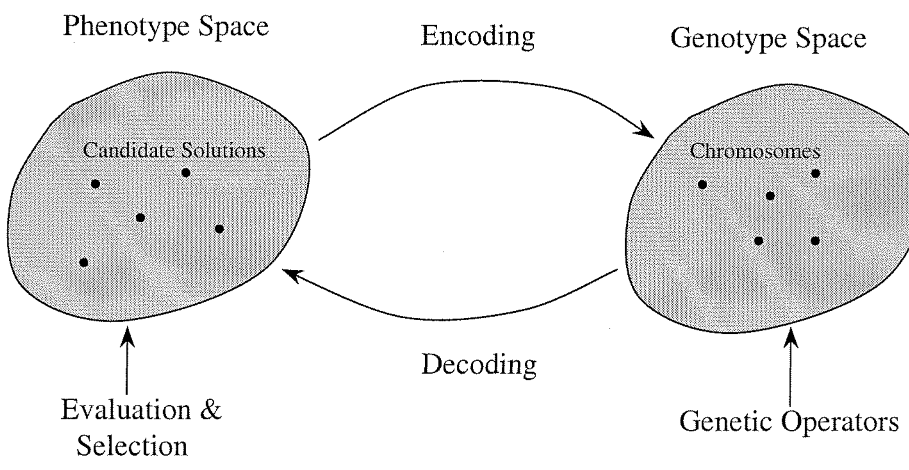


Figure 2.2: The relation of phenotype space and genotype space

for GAs. In all of Holland's work [53], chromosomes are binary strings – lists of 0's and 1's. For example, if the problem is to maximize a function of three variables, $F(x, y, z)$, each variable may be represented by a 10-bit binary number (suitably scaled), and the chromosome would therefore consist of 30 binary bits. This type of encoding is called binary coding. Binary strings have been shown to be capable of encoding a wide variety of information, and they have been used to solve optimization problems in various domains. The properties of binary string representations for GAs have been extensively studied, and a good deal is known about the genetic operators and parameter values that work well with them.

To represent variables by binary strings, the values of the variable must be discrete or can be discretized. For example, integers can be represented by binary strings using standard binary coding or Gray coding. If the variables are actually continuous, *e.g.*, real numbers, the binary string length must be carefully chosen so that the discretization of the variables provides enough resolution to get the desired level of precision.

As Genetic Algorithms are applied to more applications in different areas, especially for the constrained optimization problems from engineering world, binary strings are found not to be a natural coding scheme. As Davis [26] points out, in such problems, the problem parameters are often numeric (real), so representing them directly as real numbers, rather than binary strings, seems obvious and concise, and may have advantages. In general for an optimization problem with k real parameters that need to be optimized, a real number coding uses a vector of k real numbers to encode the solution vector [50, 61, 119]:

$$x = (x_1, x_2, ..., x_k)$$

One advantage of real number coding over binary coding is improved precision. Another advantage is that meaningful, problem-specific genetic operators can be more easily defined. For example, for real number coding, taking the average or geometric mean of two real numbers can be taken as the crossover operator. Such genetic operators obviously are more meaningful than the operators applied to binary strings. Janilow and Michalewicz [61] made a direct comparison between binary and real number representations, and found that real coding gave faster, more consistent and more accurate results.

In addition to real number coding, a lot of other coding techniques have been explored in the last 20 years such as integer coding for combinatorial optimization problems [41], embedded lists for scheduling problems [79], matrix coding for data structure [22], and LISP S-expressions for computer program evolving [65].

An encoding scheme is the linkage between the solutions in the phenotype space and the chromosomes in the genotype space, and choosing an appropriate representation of candidate solutions to the problem at hand with corresponding genetic operators is the foundation for applying Genetic Algorithms to solve real world problems. For any application case, it is necessary to perform analysis carefully to ensure an appropriate encoding of solutions together with meaningful and problem-specific genetic operators. Although many empirical and theoretical results are available for the standard instances of Genetic Algorithms, including encoding scheme and corresponding genetic operators, most practitioners prefer natural, problem-related encoding schemes along with carefully designed, meaningful, problem-specific genetic operators. Michalewicz [79] concludes that utilizing a "natural" representation of candidate solutions for a given problem with a family of applicable genetic operators is a promising direction for problem solving in general.

2.5 Selection

The key principle of Darwinian natural evolution theory is that the fitter individuals have a greater chance to reproduce offspring, and it is by the principle of "survival of the fittest" that the species evolve into better forms. In Genetic Algorithms, the bias towards fitter individuals is achieved through selection. The objective of any selection scheme is to statistically guarantee that the fitter individuals have a higher probability of selection for reproduction. The term *selection pressure* is used to refer to the degree of bias towards fitter individuals for a selection scheme, and selection pressure is the driving force which determines the rate of convergence of a GA to the optimal solution. Low selection pressure will give the algorithm more chance to explore the search space, and high selection pressure pushes the algorithm towards the direction of the highly fit individuals, and results in exploitation of the most promising search region. Choosing an appropriate selection scheme is important, and the behavior of the GA strongly depends on the selection scheme. An inappropriate selection scheme may result in premature or slow convergence.

In a GA, selection is carried out at two different stages: parent selection and generational selection. Parent selection is the step in which individuals from the parent generation are selected as parents to create offspring. Generational selection is carried out after a specified number of offspring are generated. In general, the new generation is created by selecting individuals from both the parent generation and the offspring generation. Most of the proposed selection schemes belong to the following two categories: stochastic selection and deterministic selection. For parent selection, stochastic selections are usually applied, and for generational selection, deterministic selections are usually used. Fitness proportionate selection and tournament selection are two of the most popular stochastic selection algorithms. The next section describes fitness proportionate selection, and tournament selection is described in Appendix C.

2.5.1 Fitness Proportionate Selection

Fitness proportionate selection is typically implemented as a probabilistic operator for parent selection, using the relative fitness to determine the selection probability of an individual. In fitness proportionate selection, the selection probability for each chromosome is proportional to its fitness value. For chromosome k with fitness value F_k, its selection probability p_k is calculated as follows:

$$p_k = F_k / \sum_{j=1}^{pop\_size} F_j$$

For example, roulette wheel selection is a proportionate selection scheme in which the slots of a roulette wheel are sized according to the fitness of each individual in the population. An individual is selected by spinning the roulette wheel and noting the position of the marker. The probability of selecting an individual is therefore proportional to its fitness. Another well-known proportionate selection scheme is stochastic universal selection [9]. Stochastic universal selection is a less noisy version of roulette wheel selection in which N equi-distant markers are placed around the roulette wheel, where N is the number of individuals in the population. N individuals are selected in a single spin of the roulette wheel, and the number of copies of each individual selected is equal to the number of markers inside the corresponding slot.

Proportionate selection methods assign selection probability to an individual according to its fitness, and this can be problematic. Proportionate selection depends upon positive values, and, simply adding a large constant value to the objective function can eliminate any selection pressure, with the algorithm then proceeding as a purely random search. There are several heuristics that have been devised to compensate for these issues. Using *fitness scaling*, the fitness of all parents can be scaled relative to some reference value, and proportionate selection then assigns selection probability according to the scaled fitness values. Fitness scaling is described in detail in Appendix B. *Rank-based selection* methods utilize the indices of individuals when ordered according to fitness to calculate the corresponding selection probabilities, rather than using absolute fitness values [10]. Rank-based selection also eliminates problems with functions that have large offsets. Linear as well as nonlinear fitness mappings have been proposed for rank-based selection as illustrated in the following.

Rank-Based Selection

For rank-base selection, individuals are sorted according to their fitness values, and the selection probability of each chromosome is assigned according to its rank instead of its raw fitness. After individuals are sorted in order of raw fitness, the reproductive fitness values are assigned according to rank. Two methods for mapping rank into reproductive fitness values are in common use: linear ranking and exponential ranking.

Let F'_k be the reproductive fitness value for the k^{th} chromosome in the ranking of population; the linear ranking takes the following form:

$$F'_k = q - (k-1) \times \frac{q - q_0}{pop\_size - 1}$$

where parameter q is the reproductive fitness for the best chromosome, and q_0 is the reproductive fitness for the worst chromosome. Fitness values of intermediate chromosomes are decreased from q to q_0, proportional to their rank. When q_0 is set to 0, it provides the maximum selective pressure.

Michalewicz proposed the following *exponential ranking* method [79]:

$$F'_k = q(1-q)^{k-1}$$

A larger value of q implies stronger selective pressure. Hancock proposed the following exponential ranking method:

$$F'_k = q^{k-1}$$

where q is typically about 0.99. The best chromosome has a fitness of 1, and the last one receives q^{pop_size-1}.

Controlling the variance of the fitness values is one of the frequent problems of GA's. Ranking assures that the variance is constant throughout the optimization process. Mapping ranks to reproductive fitness for proportionate selection gives a similar result to fitness scaling, in that the ratio of the maximum to average fitness is normalized to a particular value. However, it also ensures that the remapped fitnesses of intermediate individuals are regularly spread out. Because of this, the effect of one or two extreme individuals will be negligible, irrespective of how much greater or less their fitnesses are than the rest of the population.

2.5.2 Deterministic Selection

Deterministic selection schemes select individuals according to their fitness values in a deterministic way.

Deterministic selection schemes are usually used in generational selection to select individuals from both the parent generation and offspring generation to create the next generation. An example of deterministic selection is the *generational replacement* which replaces the entire parent generation by their offspring (*i.e.*, the offspring generation is taken as the new generation, and the parent generation is discarded after the offspring generation is created).

Elitist selection [31] is another popular deterministic selection scheme which ensures that the best chromosome is passed onto the new generation if it is not selected through another process of selection. Fitness proportionate selection does not guarantee the selection of any particular individual, including the fittest. Thus with fitness proportionate selection the best solution to the problem discovered so far can be regularly thrown away. Sometimes this is counterproductive. For many applications the search speed can be greatly improved by not losing the best, or *elite*, member between generations. Ensuring the propagation of the elite member is termed *elitism* and by retaining the best chromosome in the population, elitist selection guarantees asymptotic convergence.

The (μ, λ)-evolution strategy [6] uses a deterministic selection scheme which has been introduced to Genetic Algorithms. The notation (μ, λ) indicates that μ parents create $\lambda > \mu$ offspring by means of recombination and mutation, and the best μ offspring individuals are deterministically selected to replace the parents. Notice that this mechanism allows that the best member of the population at generation $t + 1$ might perform *worse* than the best individual at generation t (*i.e.*, the method is not *elitist*), thus allowing the strategy to accept temporary deteriorations that might help to leave the region of attraction of a local optimum and reach a better optimum. In contrast, the $(\mu + \lambda)$-evolution strategy [7] selects the μ survivors from the union of parents and offspring, such that a monotonic course of evolution is guaranteed.

2.6 Crossover

Once two chromosomes are selected, the crossover operator is applied to generate two offspring. Crossover combines the schemata or building blocks from two different solutions in various combinations (see Appendix E for more information about schemata and building blocks). Shorter good building blocks are converted into progressivly longer good building blocks over time until an entire good solution is found. Since highly fit individuals are more likely to be selected as parents, the GA examines more candidate solutions in good regions of the search space and fewer candidate solutions in other regions. Crossover is the most important genetic operator for a GA, and it is the driving force for exploration of the search space to find a optimum solution. The performance of the GA depends to a great extent on the performance of the crossover operator used. How crossover works can be explained by schema theorem [53] which is detailed in Appendix E.

The implementation of crossover depends on the encoding scheme used. Most canonical GAs use binary coding, and therefore the crossover schemes for binary codings (called binary crossover here) have been studied most thoroughly. Several real crossover schemes also have been proposed and applied to optimization problems utilizing real number encoding. These crossover operators are discussed in Appendix D, and one particular real crossover scheme, blend crossover, is described in the following.

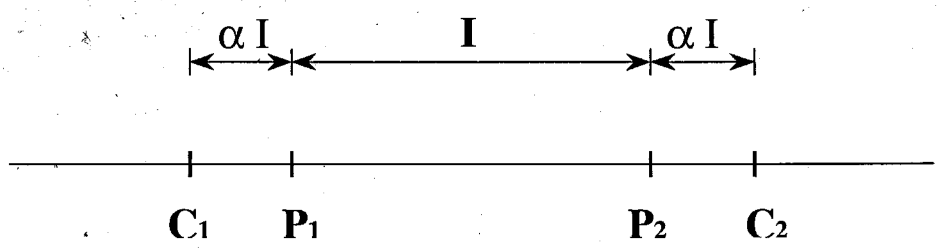


Figure 2.3: Blend crossover (BLX-α)

Blend Crossover

Random crossovers are one kind of real crossover that essentially create offspring randomly within a hyper-rectangle defined by the parent points. *Flat crossover* is a basic random crossover given by Radcliffe [95], which produces an offspring by uniformly picking a value for each gene from the range formed by the values of two corresponding parents' genes. A generalized crossover, called *blend crossover* (denoted as BLX-α), is proposed in [33] to introduce more variance into the operator. BLX-α also uniformly picks values from a range, but this range is not formed by the values of the two parents' genes. As illustrated by Figure 2.3, P_1 and P_2, the gene values (real number) of two parents, form the interval I, and α is a user-defined real parameter. BLX-α first extends interval I with αI on both sides such that the *crossover interval* is limited by points C_1 and C_2. Then a point is randomly picked from the crossover interval. By extending the interval formed by the parents to include regions beyond the parents, blend crossover introduces more exploration into the search, and often increases the robustness of the algorithm. Also, since the area of the extension is proportional to the area of the interval formed by the parents, the amount of exploration introduced is actually determined by the closeness of the parents. As the evolution proceeds, chromosomes in the population become converged, and the parents become similar, and the disruptive exploration introduced by blend crossover becomes less significant. The parameter α controls the amount of disruptive exploration, and it is usually set as 0.5, such that the probability that an offspring lies outside its parents is equal to the probability that it lies between them.

2.7 Mutation

After new individuals are generated through crossover, mutation is applied with a low probability to introduce random changes into the population. In a binary-coded GA, mutation may be done by flipping a bit of a binary string, as shown in Figure 2.4, while in a nonbinary-coded GA, mutation

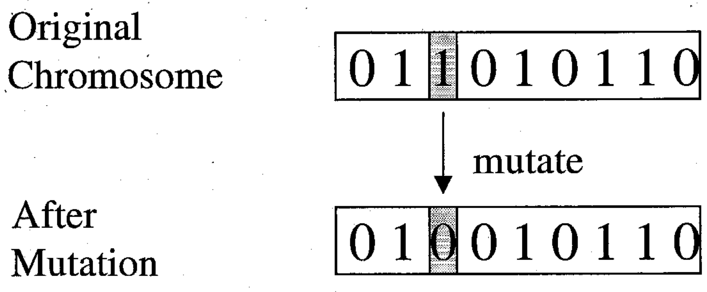


Figure 2.4: Mutation

involves randomly generating a new value in a specified position in the chromosome. In GAs, mutation serves the crucial role of replacing the gene values lost from the population during the selection process so that they can be tried in the new context, and of providing the gene values that were not present in the initial population. By introducing random changes into the population, more regions of the search space can be evaluated, and premature convergence can be avoided.

For example, in a binary-coded GA, consider a particular bit position, say bit 10, which has the same value (0) for all chromosomes in the population. In such a case, crossover alone cannot explore new solutions since crossover does not generate a new gene value, *i.e.*, crossover cannot create a chromosome with a value of 1 for bit 10, since its value is 0 in all parents. If a value of 0 for bit 10 turns out to be suboptimal, then, without mutation, the algorithm will have no chance of finding the best solution. However, if mutation is applied, there will be some probability that value 1 will be introduced in bit 10 of some chromosomes, and if this results in an improvement in fitness, those chromosomes will be multiplied by the selection algorithm and the value 1 at bit 10 will be inherited by other offspring through crossover. Although the crossover operator is the most efficient and important genetic operator, by itself, it does not guarantee the reachability of the entire search space with a finite population size. It is mutation that makes the entire search space reachable.

In real encoding, the basic mutation operator, called *uniform mutation*, simply replaces a gene (real number) with a randomly selected real number within a specified range. Other real mutation schemes have been developed. For example, *boundary mutation* replaces a gene with either the lower bound or the upper bound [80]. A direction-based mutation was presented in [46]:

$$x' = x + r \cdot d$$

where r is a randomly generated nonnegative real number, and d is the approximate gradient direction vector of the objective function f with its k^{th} component defined as

$$d_k = \frac{f(x_1, ..., x_k + \Delta x_i, ..., x_n) - f(x_1, ..., x_k, ..., x_n)}{\Delta x_k}$$

More details about implementations of real coded mutation operators can be found in [44].

The mutation probability P_M is defined as the probability of mutating each gene. It controls the rate at which new gene values are introduced into the population. In general, mutation is implemented as a background operator, and is applied with low probability. If the mutation probability is too high, too many random perturbations will occur, and the offspring will lose their resemblance to their parents. The ability of the algorithm to build on the history of the search will then be lost.

2.8 Example

In this section, a Genetic Algorithm is constructed to search for the maximum of a multi-modal function. Implementation details and the optimization result are shown to illustrate how to construct the various components of a GA and the characteristics of optimization using GAs.

The optimization problem is stated as follows: find the maximum of function

$$f(x) = x\sin(\pi x), \quad 0 \leq x \leq 6$$

As shown in Figure 2.5, function $f(x)$ is multi-modal, and the global maximum is at $x = 4.522$, where $f(4.522) = 4.511$.

To construct the Genetic Algorithm for this optimization problem, real number encoding is applied. A real number is used to represent variable x. The blend crossover BLX-α and uniform mutation for real number encoding are applied as the genetic operators. Since the problem here is to search for the maximum of function $f(x)$, function $f(x)$ will serve as the evaluation function, and for each candidate solution x, the function value $f(x)$ is used as the fitness value. Linear ranking selection scheme is utilized to select individuals from the parent population to go through reproduction, and elitist selection is applied to select the best individuals from the parent population and the offspring population to form the next generation.

In the test run, the population size is set to 10, and the maximum number of iterations is set to 10. Figure 2.6 shows the evolution of the distribution of the candidate solutions in each generation as the iteration proceeds. For each generation (iteration), the distribution of the 10 candidate solutions

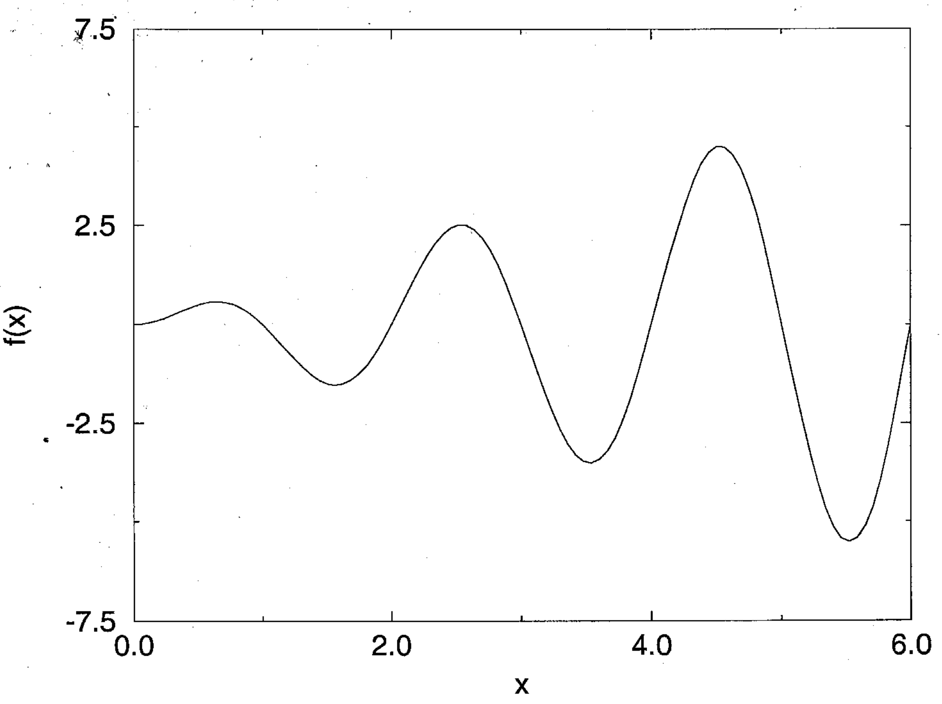


Figure 2.5: Function $f(x) = x\sin(\pi x)$

and their fitness values are shown by the symbols on the plot of $f(x)$. It's clear that as the iteration proceeds, the exploration effort is gradually concentrating on the area around the global maximum. Figure 2.7 shows the convergence of the average fitness values in each iteration to the maximum value of the function: 4.511.

2.9 Summary

Genetic Algorithms are stochastic optimization techniques that are based on the adaptive mechanics of natural selection evolution. They have been used to solve many optimization problems where the fitness surface is nonlinear, multi-modal, and dependent on several parameters simultaneously. The general structure and the main components, encoding, selection, crossover, and mutation, was described in this chapter, and different schemes for each component have been introduced. Although proper problem-specific encoding scheme and genetic operations need to be developed for a specific optimization problem, the general schemes provide guidelines for development of a specific implementation of GA. At the end of the chapter, a GA was constructed to search for the maximum of a multi-modal function. Implementation details and the optimization result were shown to illustrate how to construct the various components of a GA and the characteristics of optimization using

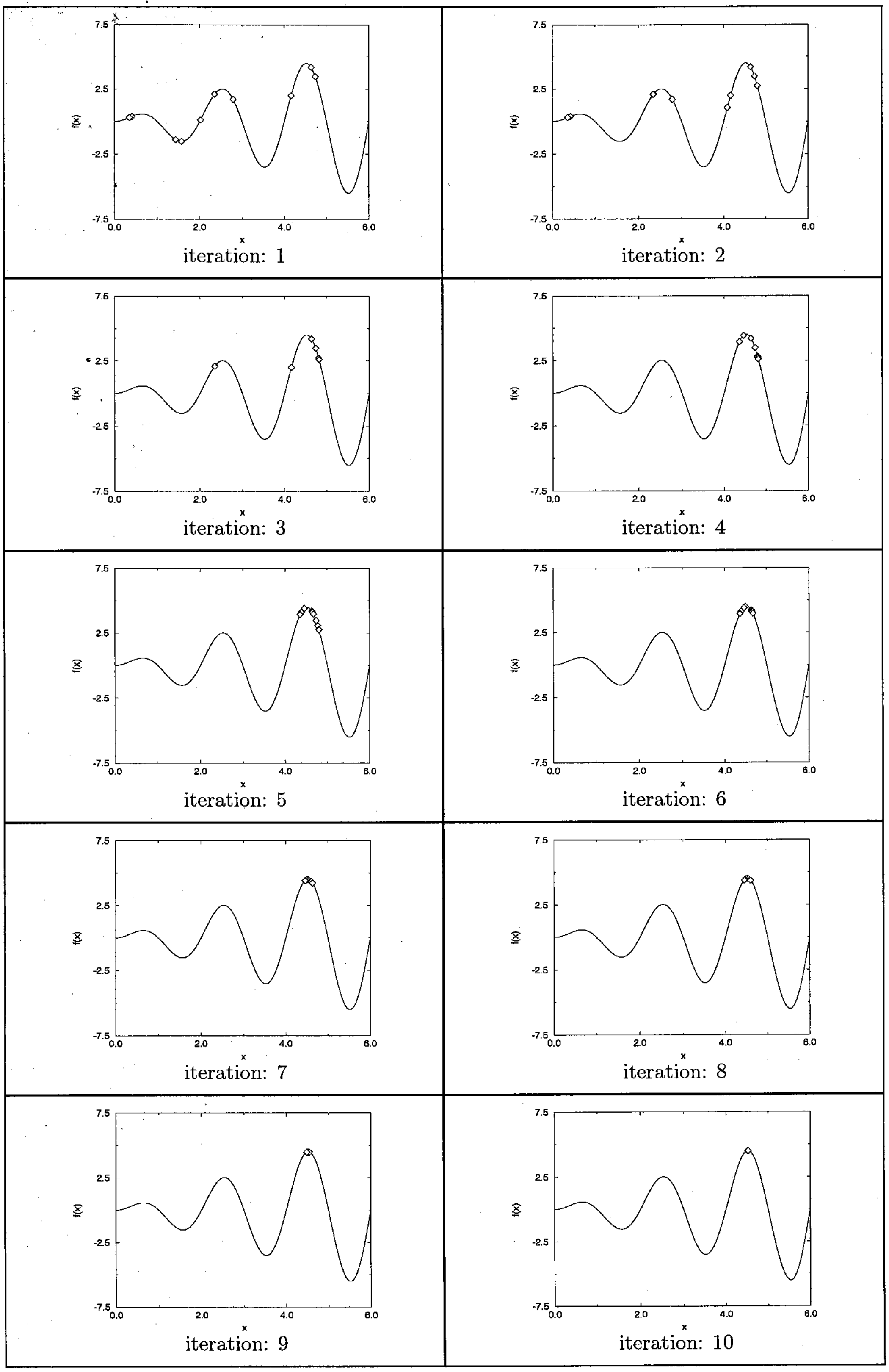


Figure 2.6: Results of the GA optimization

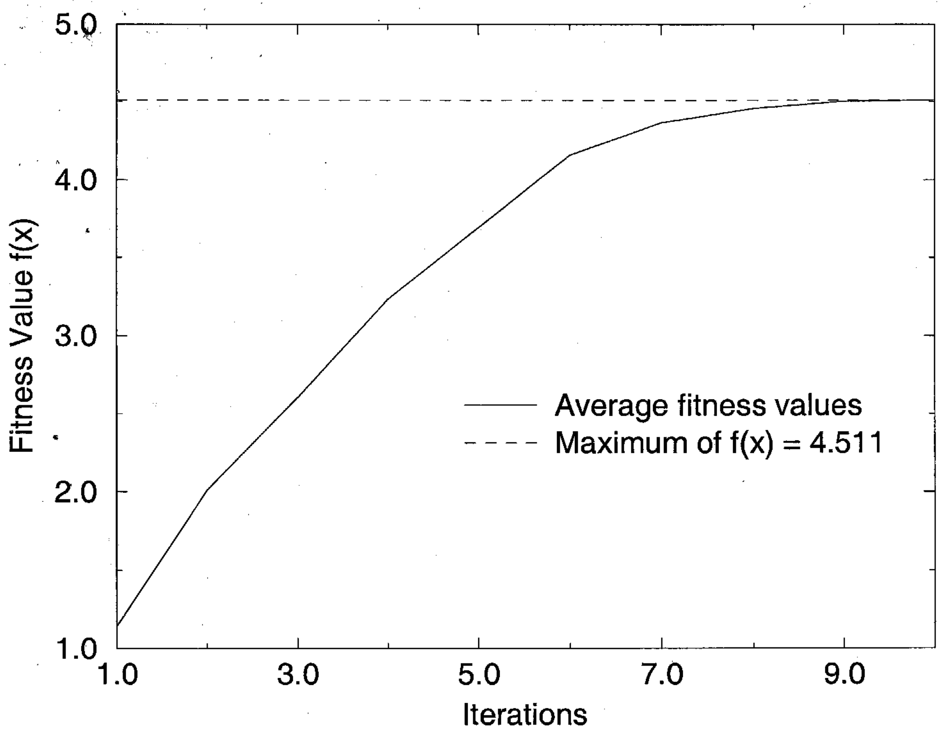


Figure 2.7: Convergence of the average fitness values

GAs.

Chapter 3

Bulk Wet Etching Simulation Tool

Since the approach developed here for mask-layout and process synthesis problem is to use simulation of fabrication process in an iterative refinement loop, a fabrication process simulation tool that is both efficient and accurate is important to the implementation of the method. This chapter will briefly review a bulk wet etching simulation tool called SEGS[55], which is computational efficient and geometrically accurate.

The bulk wet etching process is a widely used technique to inexpensively produce high aspect ratio 3-D mechanical structures with the use of chemical etchants. Most of the chemicals used today are anisotropic etchants which attack different crystallographic orientations of the material at different rates and provide control over the intended 3-D shape of the etched wafer. All bulk wet etching simulation programs predict the three-dimensional geometry change of the underlying etching structure for different etching time steps based on the input of a mask-layout and etch rate data. Early etch modeling work includes the Wulff-Jaccodine method of traveling planes [42, 59]. A number of computer implementations have been developed [17, 18, 28, 29, 107, 108, 115]. Another important etching simulation tool is the Slowness method by Foote and Sequin [106]. The models of the Wulff-Jaccodine method and the Slowness method are purely based on geometric analysis, while Hubbard and Antonsson [56], Than and Buttgenbach [114] developed cellular automata (CA) algorithms which base the rate of removal of each cubic cell of material on the number of nearest neighbor cells remaining. The geometric analysis is fast and accurate but is limited to basic shapes; the cellular automata approach is slow but is able to simulate the etching of shapes of any complexity.

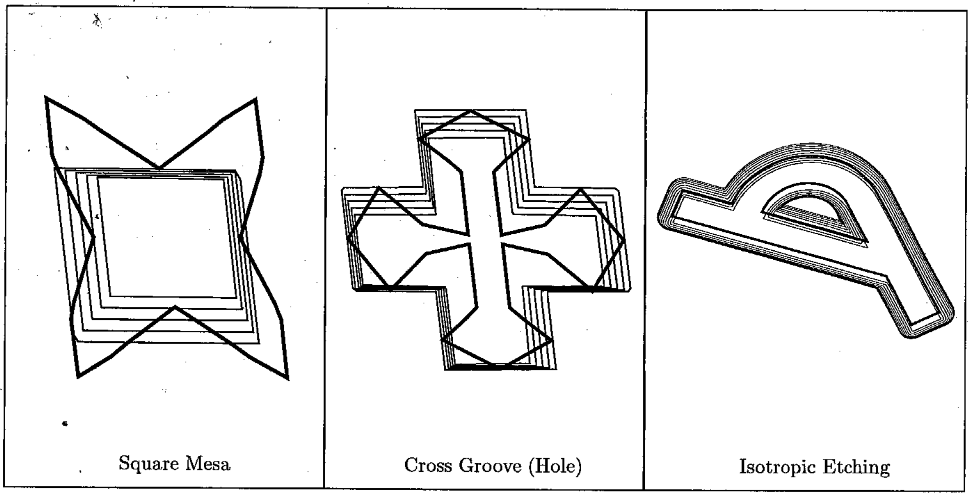


Figure 3.1: Examples of SEGS wet etching simulation

SEGS Simulation Tool

SEGS is a bulk wet etching simulation tool developed by Hubbard and Antonsson [55]. SEGS uses an edge segmentation approach to trade-off the advantages of the geometric analysis and cellular automata. The basic approach is to start with the polygonal mask-layout boundary of a vector method, then subdivide each straight line segment into many smaller segments, and these small segments are updated for further time steps, and local and global intersection are easily identified and handled. The input to the simulation tool is the mask-layout, etch rate data, etch time, and some other computation parameters, and the output of the simulation is a series of polygon layers (contours) at different depths, with each contour representing the cross section of the etched structure at that particular depth. Figure 3.1 shows examples of etched shapes simulated by SEGS. The black polygons are the mask-layouts, and the etched shapes are represented by a series of polygon layers.

Originally SEGS can only simulate a single etching step where the initial wafer surface is flat. Although single step anisotropic wet etching has long been used for fabricating microstructures such as diaphragms, V-grooves, and cantilevers on a silicon wafer, the shapes of the microstructures fabricated by this etching system are limited, because a conventional single-step process provides only a limited set of crystallographic planes appearing on the etching profile. Various groups have proposed using multi-step chemical anisotropic etching process in order to achieve more complex three-dimensional microstructures on single-crystal silicon [43, 63, 98, 109]. Using several different etchants in a fabrication makes it possible to fabricate shapes which cannot be created using single etchant, and the range of shapes that can be produced is greatly expanded. Designing the multi-steps

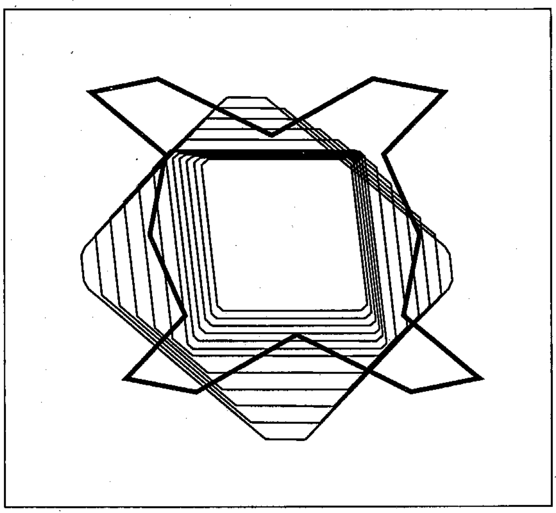


Figure 3.2: Simulated result of a two-step wet etching fabrication

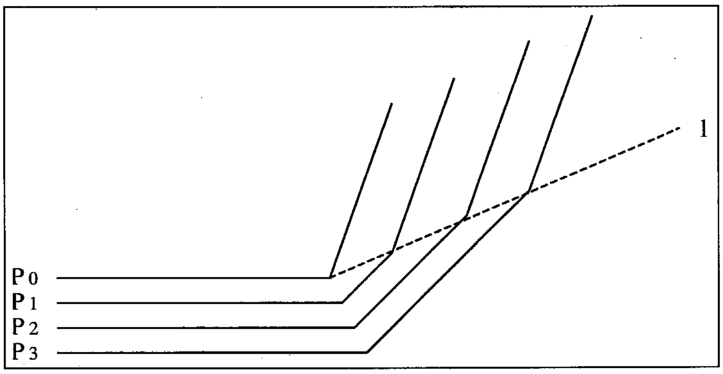


Figure 3.3: Multi-step etching simulation: Moving of interface between old wall and new wall (side view)

of an etching process requires an etching simulator.

SEGS was further extended to handle simulation of multi-step wet etching fabrication. The fabrication of micro structure using two different etchants in sequence can be simulated. Figure 3.2 shows a structure which results from the simulation of a two-step wet etching fabrication. The cross sections of the etched structure are shown, and the black polygon is the mask-layout. Using this simulation system, it is possible to design multi-steps into the anisotropic etching process which provides a variety of three-dimensional microstructures. An synthesis example for a two-step anisotropic etching process will be described in Chapter 6.

In extending the function of SEGS to handle multi-step etching simulation, the major difficulty is in the calculation of the moving of the interface between the walls produced by different applied etchants, since the walls produced by different etchants usually have different slope angles. Figure 3.3 shows how the structure profile changes when the second etchant is applied to the already existed structure resulting from the etching using the first etchant. Each profile shows the etched structure at a different etch time step. Profile p_0 shows the structure at the time t_0 when the second etchant is just applied. Profile p_1 shows the structure at time $t_0 + \delta t$, when a new wall with different slope angle appears at the corner. As the etching proceeds, the interface between the new wall and the wall resulting from the first etchant is moving up (line l). Since SEGS calculates and maintains the etching data by using contours at fixed discrete depths, the moving of the interface of walls must be handled carefully.

SEGS has been used to simulate a wide variety of mask-layouts, and its accuracy has been tested through experimental verification. SEGS proved to be stable and fast. To simulate shapes shown in Figure 3.1 on a Sun Ultra10 workstation with a 440 MHz CPU clock speed, a typical simulation time is only several seconds.

Surface Reconstruction and Visualization

Although SEGS is efficient and accurate in predicting the fabricated device geometry, it represents the simulated device shape using contours at different depths, not as surfaces. Contours are not amenable to the analysis and design of micro systems, since it is often necessary to determine the object interaction and mechanics in which surface or solid representation of shape is required. Work has been done to achieve surface reconstruction from the contours via Delaunay Triangulation and visualization of the reconstructed surfaces. Figure 3.4 shows the reconstructed surfaces of the device shape that has etching contours shown in Figure 3.2. Interested readers are referred to Lee and Antonsson [67] for details about the algorithm for the surface reconstruction.

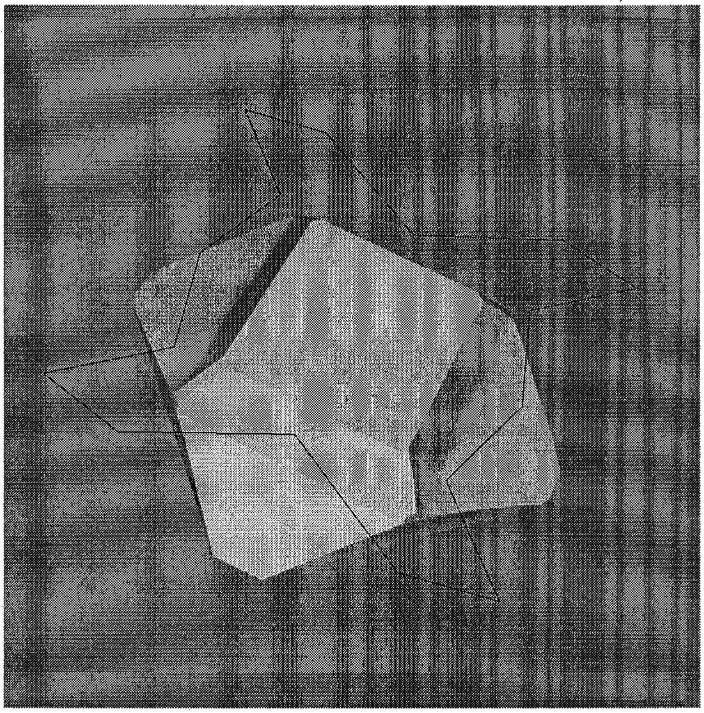


Figure 3.4: Visualization of the reconstructed surface of etching contours

Chapter 4

Shape Matching

4.1 Introduction

The approach developed here to the MEMS mask-layout and process synthesis problem is to use Genetic Algorithms as an optimization method combining with a simulation tool, to search for the optimum solution. As discussed in Chapter 2, one of the key issues in applying Genetic Algorithms to solve a practical problem is that there must be an evaluation function (or performance function). The evaluation function evaluates candidate solutions to determine the fitness value for each solution so that the genetic operators can manipulate the solution population accordingly to find the optimum solution. For the mask-layout and process flow synthesis problem, the candidate solutions in the solution space are mask-layouts and some process parameters. For each candidate solution, simulation is performed using the mask-layout and process parameters specified by the candidate solution, and a simulated device shape is obtained. If the simulated shape is geometrically very close to the desired target shape, the candidate solution (the mask-layout and the process flow) will be considered to be fit (with a high fitness value). The fitness value for each candidate solution should depend on how close the simulated shape is to the target shape. The simulation program can be viewed as part of the evaluation function, but more is needed to assign a fitness value to each candidate solution. A shape matching scheme is needed to compare the simulated shapes with the desired, pre-specified target shape so that a value indicating the closeness (similarity measure) between a simulated shape and the target shape can be obtained and assigned as fitness value for each candidate solution.

Shape matching is a problem of both theoretical and practical importance in computer vision, and different shape matching algorithms have been developed [62, 83]. Generally speaking, comparing arbitrary shapes in three dimensions is difficult [14, 15], and most of the work is done for 2-D shape matching. Among the different shape matching algorithms, one particular category is the polygonal shape matching [2, 5, 24, 90]. Since the simulation tool SEGS used in the mask-layout and process synthesis problem represents the simulated 3-D shapes with layers of polygons, a polygonal shape matching scheme is suitable for evaluating solutions. Considering the layer representation of 3-D shapes, it is natural to decompose the 3-D shape matching into 2-D shape matching at all layers and construct the 3-D closeness metric (match value) from the 2-D polygonal match values. In the following sections, a polygonal shape matching scheme is described, and the scheme to calculate the 3-D match value using the 2-D polygonal shape match values at all layers is discussed.

Before describing the matching schemes, a note about the terminology should be made. In the following sections, Term "mismatch" will be used to describe the dissimilarity between two shapes, because the shape matching scheme quantifies the difference (dissimilarity) between two shapes.

Clarification is also needed about the term "shape." Generally speaking, for a polygon, its size and orientations and relative locations of its edges are part of the shape of the geometry. However, the polygonal shape matching scheme introduced here only considers the "pure shape" of the polygons, which is invariant to scaling. In other words, the "pure shape" of a polygon is unchanged after scaling. When "shape" is used in shape matching or shape mismatch, it may refer to the geometry's shape in general, including its pure shape and size, or to the geometry's pure shape only, invariant to scaling. The meaning should be clear from the context.

4.2 Polygonal Shape Matching

Several polygonal shape matching schemes have been proposed, and a method introduced by Arkin *et al.* [3] is described here.

Representation of Polygons

The boundary of a simple polygon A can be represented by the *turning function* $\Theta(s)$, as shown in Figure 4.1. The turning function $\Theta(s)$ measures the angle of the counterclockwise tangent as a function of the arc length s, measured from some reference point on the boundary of A. Thus $\Theta(0)$ is the angle θ_0 that the tangent at the reference point makes with some reference orientation associated with the polygon (such as the x-axis). $\Theta(s)$ keeps track of the turning that takes place, increasing with left-handed turns and decreasing with right-handed turns. Formally, if $k(s)$ is the

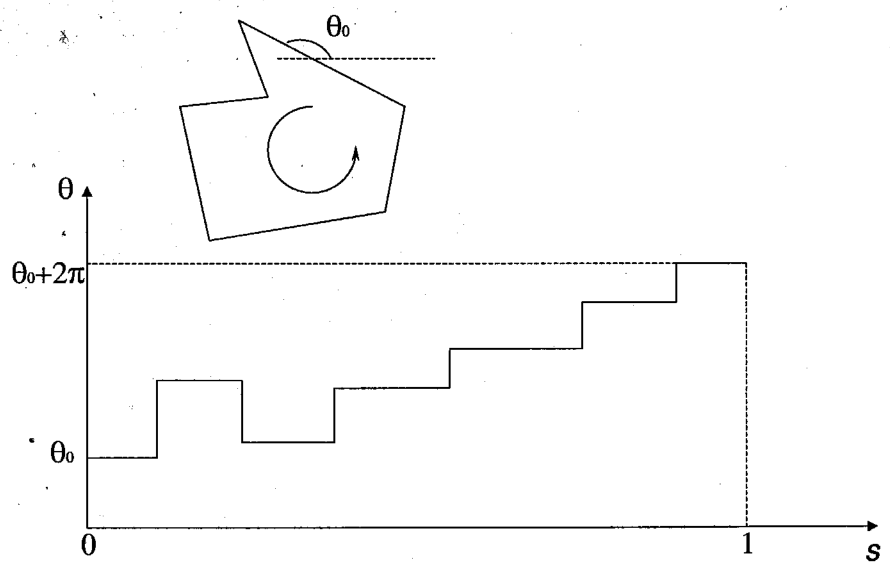


Figure 4.1: Turning function of polygons

curvature function for a curve, then $k(s) = \Theta'(s)$.

For the purpose of polygonal shape matching, each polygon is rescaled so that the total perimeter length is 1; hence $\Theta(s)$ is a function from [0, 1] to R. For a convex polygon, $\Theta(s)$ is a monotone function, starting at some value θ_0 and increasing to $\theta + 2\pi$. For a nonconvex polygon, $\Theta(s)$ may become arbitrarily large, since it accumulates the total amount of turn, which can grow as a polygon "spirals" inward. Although $\Theta(s)$ may become very large over the interval $s \in [0, 1]$, in order for the function to represent a simple closed curve, $\Theta(1)$ must be $\Theta(0) + 2\pi$ (assuming that the reference point is placed at a differentiable point along the curve). When the path is polygonal, the turning function is piecewise-constant, with jump points corresponding to the vertices of the polygon. Note that the domain of $\Theta(s)$ can be extended to the entire real line in a natural way by allowing angles to continue to accumulate as the traversal around the perimeter of the polygon continues.

The function $\Theta(s)$ has several properties which make it especially suitable for the shape matching purpose. It is piecewise-constant for polygons, making computations particularly easy and fast. By definition, the function $\Theta(s)$ is invariant under translation and scaling of the polygon. Rotation of polygon corresponds to a simple shift of $\Theta(s)$ in the θ direction. Note also that changing the location of the reference point by an amount $t \in [0, 1]$ along the perimeter of polygon corresponds to a horizontal shift of the function $\Theta(s)$ and is simple to compute (the new turning function is given by $\Theta(s + t)$).

Distance Function

The shape mismatch between two polygons is calculated by a distance function. The distance function between two polygons A and B is formally defined as the L_2 distance between their two turning functions $\Theta_A(s)$ and $\Theta_B(s)$, minimized with respect to vertical and horizontal shifts of the turning functions (in other words, minimize with respect to rotation and choice of reference points):

$$d_2(A,B) = \min_{\Theta \in R, t\in[0,1]} \left(\int_0^1 |\Theta_A(s+t) - \Theta_B(s) + \theta|^2 ds\right)^{\frac{1}{2}} \tag{4.1}$$

The above equation of distance function can be explained as follows. First the reference point and orientation of polygon B is fixed, and turning function $\Theta_B(s)$ is obtained. For polygon A, the reference point is shifted along the boundary by an amount t, and the polygon is rotated by angle θ, and the new turning function is given by $\Theta_A(s+t)+\theta$. Then the L_2 metric between the two turning functions is calculated as $(\int_0^1 |\Theta_A(s+t) - \Theta_B(s) + \theta|^2 ds)^{\frac{1}{2}}$. Finally, the minimum of the L_2 metrics over all shifts t and rotations θ is taken as the distance function between polygons A and B.

The reasons to use L_2 norm among all the L_p norms are given in reference [3], and one of the reasons is that the calculation of L_2 metric for polygons is easy and fast. Although the distance function is given as a minimum over all t and θ, it has been shown that for an m-vertex polygon and an n-vertex polygon, the distance function always reaches its minimum when t is at one of the mn discrete values on [0, 1] when one vertex of polygon A and one vertex of polygon B coincide on the s-axis (having the same arc length). Also for a given t it has been shown that the optimal θ can be calculated as a function of t, and the minimization problem for the distance function is really a one-variable (t) minimization problem. The runtime for comparing an m-vertex polygon and an n-vertex polygon is $O(mn\log(mn))$ [3].

For the mask-layout and synthesis problem, the shapes that need to be compared are etched geometries, and the orientation of the polygon contour is considered as part of the polygonal features. For this purpose, the distance function should not be rotationally invariant, and equation (4.1) should be modified to be

$$d_2(A,B) = \min_{t\in[0,1]} \left(\int_0^1 |\Theta_A(s+t) - \Theta_B(s)|^2 ds\right)^{\frac{1}{2}} \tag{4.2}$$

Size Matching

The shape matching scheme described above measures only the difference of "pure shapes" of two polygons, i.e., the distance function is invariant to scaling, and the difference of the sizes of two polygons is not considered. But for the mask-layout and process synthesis problem the size of the device geometry is an important factor. The synthesis result is not optimal if the simulated device has a different size from what is desired, even if the shape of the simulated device is perfectly identical to the desired shape. Obviously for the problem here the shape matching scheme is not enough for an overall comparison between two shapes which includes both shape and size comparisons. A size matching scheme is needed. Here the ratio between the perimeter lengths of the two polygons is used as a size matching measurement. If l_A and l_B are the perimeter lengths for polygons A and B, the size mismatch of A and B is defined as

$$Max(l_A/l_B, l_B/l_A) - 1 \tag{4.3}$$

with a mismatch of zero meaning perfect size match. Using the shape matching scheme and size matching measurement, two polygons are perfectly matched (identical) if and only if both the shape mismatch value and the size mismatch value are zero.

4.3 3-D Shape Matching

The 3-D shapes in the synthesis problem here are represented by layers of 2-D polygon contours. Now that matching methods for polygonal shape and size comparison have been developed, a scheme to construct the 3-D shape matching measurement using the 2-D polygonal matching measurements is needed.

As described in Chapter 3, the 3-D shapes are represented by layers of polygons. Figure 4.2 shows two 3-D shapes (one simulated shape, one desired target shape) represented by layers of polygonal contours. The etching simulation program ensures that the polygon layers of both shapes vertically match, and the shape mismatch (m_{shape}) and size mismatch (m_{size}) values are calculated for each pair of corresponding polygons. The overall shape match of these two 3-D shapes is constructed as follows.

The mismatch between two 3-D shapes is decomposed into two components: pure shape mismatch (M_{shape}) and size mismatch (M_{size}). Both mismatch values can be constructed as a weighted

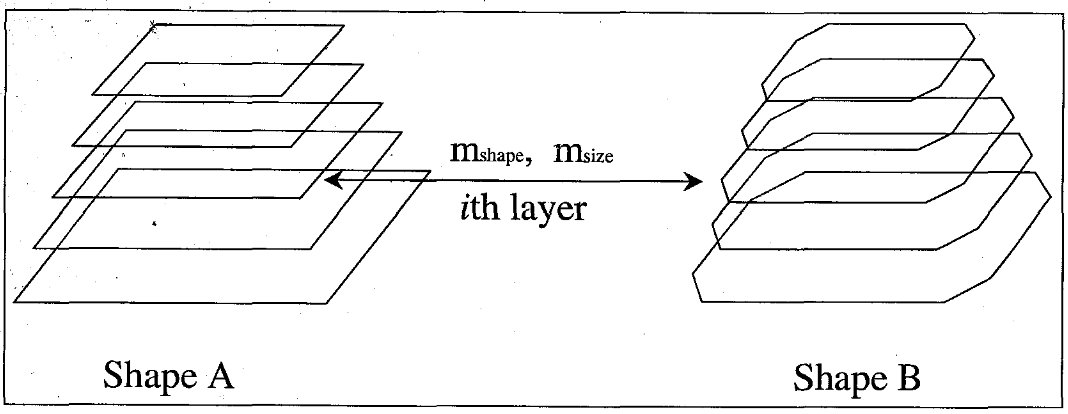


Figure 4.2: 3-D shape representation and match

sum of all shape or size mismatch values between the two polygon layers in each vertical level:

$$M_{shape} = \sum_{i=1}^{n} w_i m_{shape}(i) \tag{4.4}$$

$$M_{size} = \sum_{i=1}^{n} w_i m_{size}(i) \tag{4.5}$$

where $m_{shape}(i)$ and $m_{size}(i)$ are the shape mismatch and size mismatch values of two polygons at the ith layer, and w_is are the weights. The weights are introduced to give freedom to specify the particular importance of the feature matching of certain layers. The average is used for general purpose when the weights are the same at all layers:

$$M_{shape} = \sum_{i=1}^{n} \frac{m_{shape}(i)}{n} \tag{4.6}$$

$$M_{size} = \sum_{i=1}^{n} \frac{m_{size}(i)}{n} \tag{4.7}$$

It has been shown that two 2-D polygons with shape mismatch and size mismatch values of zero are identical. For 3-D shapes, although shape mismatch and size mismatch are also used to describe the matching between two 3-D shapes, it should be noted that not all information about the shape difference is represented by the shape mismatch and size mismatch of 2-D contours. Two 3-D shapes with shape mismatch value and size mismatch value of zero can actually be in different

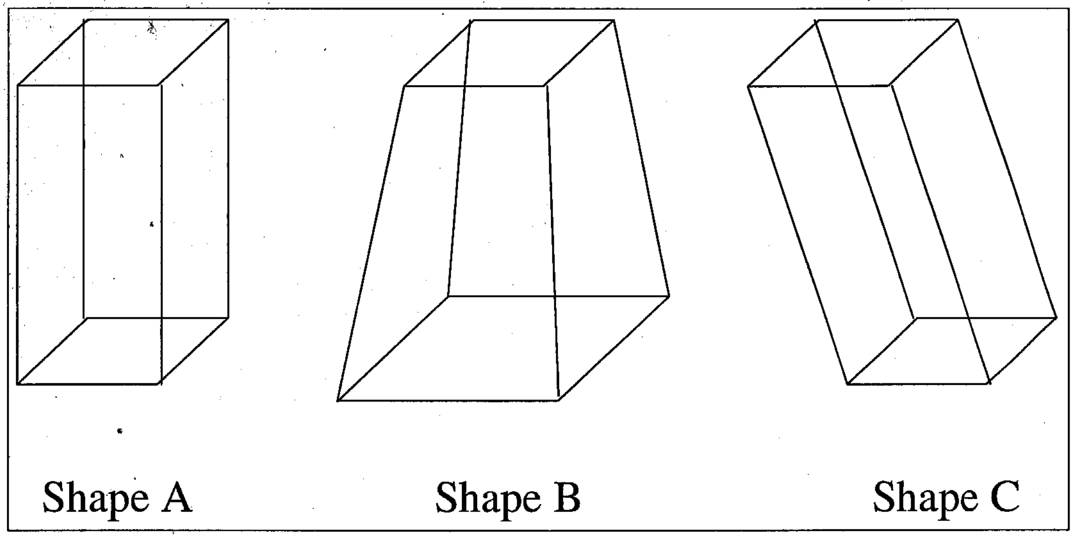


Figure 4.3: A shape mismatch of zero doesn't mean same shape

shapes in some special cases. This is because the shape mismatch M_{shape} constructed as a weighted sum of all shape mismatch values between the two polygon layers in each vertical level, as shown in Equation (4.4), does not contain all the shape information. The information about the relative position of all vertical levels is missing, because the shape matching scheme for 2-D polygons used here does not consider the positions of the polygons and is translation invariant. For two 3-D shapes with 2-D contours of the same shape (pure shape) at every level, a shape mismatch of zero will be produced using Equation (4.4), but these 3-D shapes can actually be in different shapes. For example, Figure 4.3 shows three 3-D shapes, all with cross sections of square. When comparing shape A and B, B and C, or A and C, m_{shape} for each layer will be zero because all the cross sections are in the same shape, a square. So the overall shape (pure shape) mismatch value M_{shape} will be zero. But obviously A, B and C are in different shapes. Shape A and B can still be differentiated by size mismatch value M_{size}, because m_{size} values at each layer are not all zero. However, for shape A and C, M_{size} will also be zero because all m_{size} values are zero at each layer. To differentiate shapes like shapes A and C, other measures such as the gravity center line (the line connecting centers of gravity of all cross-section polygons) may be needed.

As shown later, the experiments of the mask-layout and process synthesis are conducted on quadri-symmetric structures because of computational considerations. For quadri-symmetric structures, the center line goes straight down and is not skewed to any direction, and shapes like C in Figure 4.3 will not happen. Shape mismatch M_{shape} and size mismatch M_{size} are sufficient to serve as the shape matching evaluation parameter.

4.4 Calculating Fitness Value

In Genetic Algorithms, the performance evaluation function should ultimately assign a fitness value for each candidate solution. For the mask-layout and process synthesis problem, using the shape matching algorithm described above, naturally the candidate solution with small shape mismatch should be assigned with a high fitness value. Because a linear ranking selection scheme is used (see section 5.3 for detail), the absolute values of the fitness are not important since only the rankings of the fitness values will be used. In that sense, any algorithm can be used to convert the shape mismatch to fitness value, as long as this algorithm will ensure higher fitness value for small shape mismatch. A reciprocal function can serve this purpose:

$$\text{fitness} = \frac{1}{\text{Shape Mismatch Value}} \tag{4.8}$$

But the shape matching scheme just introduced uses two preferences to measure the shape comparison result: the "pure" shape mismatch value M_{shape}, and size mismatch value M_{size}. Before using the reciprocal function to calculate the fitness value, a mechanism is needed to combine multiple preferences (attributes) into one single preference. This procedure is called *aggregation of multiple preferences*, and an *aggregation function* serves this purpose.

If μ_1 and μ_2 are two preferences, the aggregation function is in the form

$$\mu = \mathcal{P}(\mu_1, \mu_2) \tag{4.9}$$

where μ gives the overall preference. Some examples of aggregation functions include $\mathcal{P}_{Min}(\mu_1, \mu_2) = Min(\mu_1, \mu_2)$, $\mathcal{P}_{Max}(\mu_1, \mu_2) = Max(\mu_1, \mu_2)$, and $\mathcal{P}_{\Pi}(\mu_1, \mu_2) = \sqrt{\mu_1 \mu_2}$. For aggregation functions, the term "compensation" is used to describe the effect that high performance on one attribute partly compensates for lower performance of another. $\mathcal{P}_{max}$ has highest compensation level since the overall performance is dictated by the highest-performing attribute and the performance of the other attribute has no effect on the overall performance at all. $\mathcal{P}_{Min}$ is non-compensating because the overall performance is dictated by the lowest-performing attribute. For $\mathcal{P}_{\Pi}$, the high performance on one attribute is deemed to partly compensate for the lower performance on the other. Interested readers are referred to [87] for more discussion about aggregation functions. Proper aggregation function selection is important for multi-criteria decision problems. When designing an aggregation function, it is crucial to understand the relative importance of different preferences and the way in which they interact.

Scott and Antonsson [101, 102] introduced a family of aggregation functions that spans an entire range of possible operators between $\mathcal{P}_{Min}$ and $\mathcal{P}_{Max}$:

$$\mathcal{P}_s(\mu_1, \mu_2; \omega_1, \omega_2) = \left(\frac{\omega_1 {\mu_1}^s + \omega_2 {\mu_2}^s}{\omega_1 + \omega_2}\right)^{\frac{1}{s}} \tag{4.10}$$

It is shown that

$$\begin{aligned}
\mathcal{P}_{-\infty} &= \lim_{s \to -\infty} \mathcal{P}_s &&= \text{Min} \\
\mathcal{P}_0 &= \lim_{s \to 0} \mathcal{P}_s &&= \text{geometric mean } (\mu_1^{\omega_1} \mu_2^{\omega_2})^{\frac{1}{\omega_1+\omega_2}} \\
\mathcal{P}_1 &= \lim_{s \to 1} \mathcal{P}_s &&= \text{arithmetic mean } \tfrac{\omega_1\mu_1+\omega_2\mu_2}{\omega_1+\omega_2} \\
\mathcal{P}_{\infty} &= \lim_{s \to +\infty} \mathcal{P}_s &&= \text{Max}
\end{aligned}$$

For a specific design problem, the values of s and ω_1, ω_2 should be determined by the level of compensation that should be applied. A method for establishing these values is introduced in [102]. The issues of aggregation of multiple preferences and multi-criteria decision making will be revisited in later chapters.

For the shape matching problem here, the two attributes, shape mismatch and size mismatch, are deemed to be unrelated, and the non-compensating aggregation function $\mathcal{P}_{Min}$ will be used such that only candidate solutions with good performance in both attributes are considered as good candidates. Also the discussion above about aggregation functions assumes larger preference value indicate better performance; however, in the case here, smaller shape mismatch values and size mismatch values are related to better performance, so the actual formula for shape match aggregation is *Max*, not *Min*:

$$\text{Shape Measure} = Max(M_{shape}, M_{size}) \tag{4.11}$$

and the following reciprocal function gives the fitness value:

$$\text{fitness} = \frac{1}{Max(M_{shape}, M_{size})} \tag{4.12}$$

4.5 Summary

This chapter described the shape matching algorithm for 3-D shape comparison. In the Genetic Algorithm approach to the mask-layout and process synthesis problem, the shape matching algorithm compares the simulated shapes and the target shape, and serves the performance evaluation function. Because of the polygonal layer representation of 3-D shapes used by SEGS etching simulator, polygonal shape matching scheme is used, and the overall 3-D shape matching metric is constructed by combining the shape matching values at all layers. The shape matching algorithm produces two metrics for 3-D shape matching: pure shape mismatch value, and size mismatch value. To assign a fitness value for a candidate solution in the Genetic Algorithm, an aggregation function is needed to convert the pure shape mismatch value and the size mismatch value to a single preference. Different aggregation functions are discussed and the non-compensating aggregation function $\mathcal{P}_{Max}$ is used for the problem here. Only candidate solutions with both small shape mismatch values and small size mismatch values are considered as good candidates.

Chapter 5

Mask-layout and Process Synthesis: Design and Implementation

5.1 Introduction

For the problem of synthesis of MEMS mask-layouts and processes, a Genetic Algorithm iteration loop is constructed to evolve the optimal mask and process flow as shown in Figure 5.1. The user specifies a desired 3-D target shape. An initial population of solution candidates (mask-layouts and process parameters) is randomly generated. The fabrication of each mask-layout using the associated process parameters is then simulated by a specified fabrication simulator to produce a three-dimensional shape. The performance of each solution candidate is measured through the shape comparison between the produced shape and the user defined desired shape. During each evolutionary loop, genetic operations are applied to control the stochastic searching behavior such that the best performing solution candidates are more likely to survive and evolve even better offspring for the next generation. The iteration is stopped when one satisfying solution is found.

As described in Chapter 2, for a Genetic Algorithm to work, in addition to setting some parameters, implementation of several aspects must be provided, including the coding scheme, genetic operations (crossover, mutation, *etc.*), and an evaluation of performance. For the mask-layout and process synthesis problem, each candidate solution in the solution space is a mask-layout associated with some process flow parameters. The performance evaluation of candidate solution is handled through the fabrication simulation and shape matching algorithm described in Chapter 3 and Chapter 4. To design a coding scheme and genetic operations for process parameters in general is relatively easy, and depends on the particular fabrication procedure that is considered in the synthesis. Later the implementation details of the coding scheme and genetic operations for a two-step

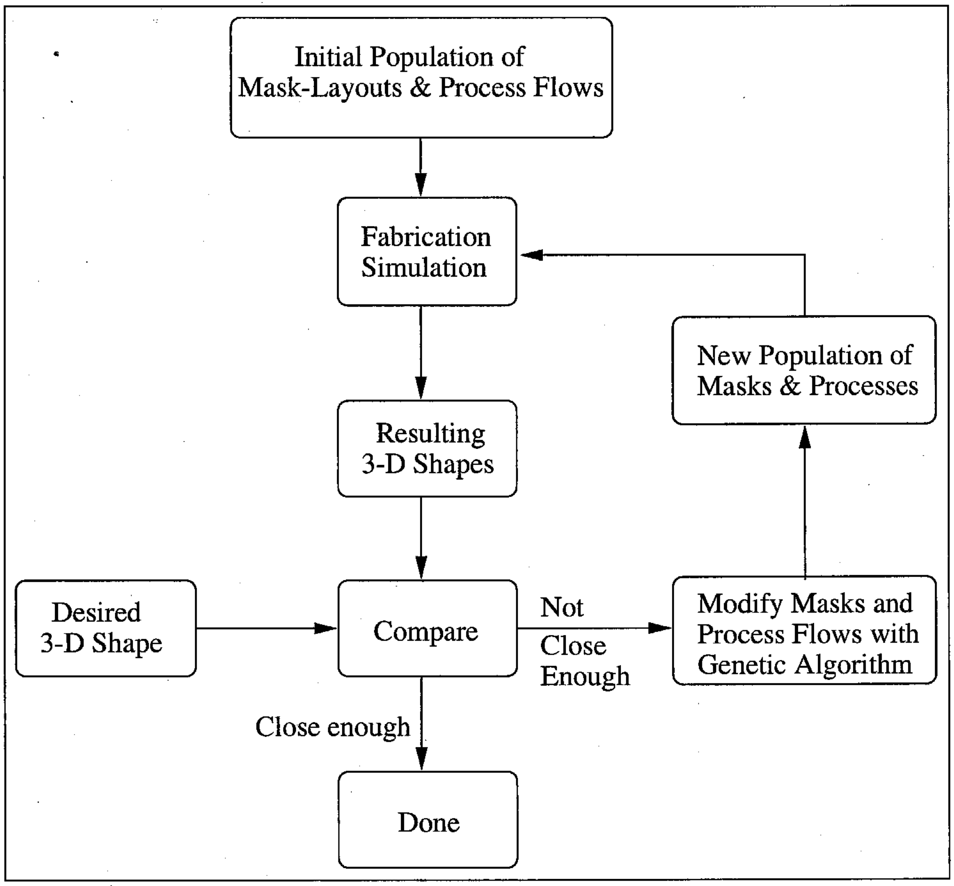

Initial Population of Mask-Layouts & Process Flows
Fabrication Simulation
New Population of Masks & Processes
Resulting 3-D Shapes
Desired 3-D Shape
Compare
Not Close Enough
Modify Masks and Process Flows with Genetic Algorithm
Close enough
Done

Figure 5.1: A schematic representation of a Genetic Algorithm MEMS synthesis technique

wet etching process will be shown, during which two different etchants will be applied to a mask-layout in sequence to create the final device shape. For mask-layouts, things are more complex, and more attention is required to design a proper coding scheme and genetic operations, which is described below.

5.2 Mask-layouts: Coding and Genetic Operations

5.2.1 Coding Scheme

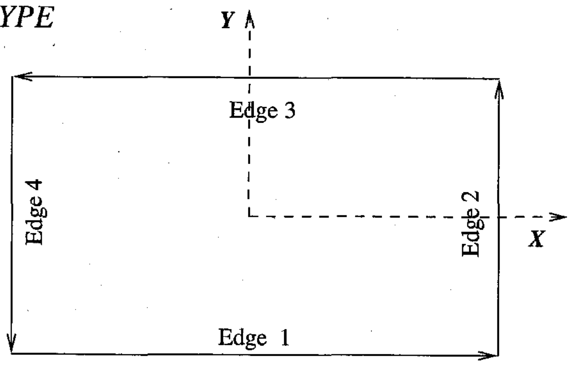
As introduced in Chapter 2, in a Genetic Algorithm, the genetic operations work on the *genotypes* (chromosomes), which are encoded strings of the candidate solutions (*phenotypes*). To design a coding scheme to encode the candidate solutions into genotypes is the first step in the design of a Genetic Algorithm.

Mask-layouts are treated geometrically as two-dimensional polygons. An appropriate coding scheme is needed to encode 2-D polygons with strings which can be easily manipulated by genetic operations. In Chapter 4, *turning function* $\Theta(s)$ is described to represent the boundary of a simple polygon. Essentially the turning function measures the length and angle of each edge of the polygon, and edges are considered as the constructing units for a polygon. Similar to turning function, for mask-layout encoding, edges are used as the constructing units, and *edge coding scheme* is developed to encode polygons with strings.

For the edge coding scheme, edge length and edge directional angle are chosen to describe each edge of a polygon [4, 72]. First a vertex on the polygon is randomly chosen, and the counterclockwise traversal of the polygon is started. For each passing edge, the length of the edge and the directional angle of the edge are recorded separately in two strings, the *distance string*, and the *angle string*. The directional angles are measures from the positive x axis in counterclockwise direction (*i.e.*, the nominal angle). In this edge coding scheme, a mask polygon is encoded into two real strings. One string contains edge directional angles and the other contains edge lengths. The size of each string is equal to the number of polygon sides. Two elements, one from each string with the same gene position, describe an edge of the polygon. The distance strings and angle strings are genotypes for the mask-layouts in the GA. A schematic illustration of the edge coding scheme is shown in Figure 5.2. Note that real strings are used in the above coding scheme. A real coding scheme is used because it provides adequate precision with a short string length.

An alternative way to encode polygons is the *vertex coding scheme*, in which the vertices are

PHENOTYPE



A Rectangle Mask Polygon With Edge Length 10 & 20

GENOTYPE

	Edge 1	Edge 2	Edge 3	Edge 4
Angle String:	0.00	1.57	3.14	4.71
Distance String:	20	10	20	10

Figure 5.2: A schematic illustration of the GA coding scheme for mask-layouts

used as the constructing units for the polygons, and the coordinates of the vertices are recorded in strings. For the mask-layout synthesis problem, the edge coding scheme has advantages over the vertex coding scheme. The advantage of using polygon edges as the constructing units instead of vertices is that direct control over the manipulation of the polygon shape is provided using the edge coding scheme. The features of a polygon shape are captured by its edges, not its vertices. The correlation between the change of an edge and the change of the polygon shape is clear, while such a clear correlation can't be achieved using vertices as the constructing units. Also, using an edge coding scheme makes designing sensible genetic operations easy and convenient as illustrated in later sections. Refer to [68] for more discussion on the advantages of edge coding schemes.

5.2.2 Mask Initialization

Genetic Algorithms work as an iterative loop, and the process to create the first generation of candidate solutions is called *initialization*. For mask-layout synthesis, the goal of the initialization process is to randomly generate the first generation of polygon shapes which uniformly sample the solution space. Different initialization procedures are tested and compared (please refer to [68] for detail), and the procedure used here is implemented in the following way.

First an integer number is randomly chosen in a user-defined range as the polygon side number by a random generator. Then using the edge-coding scheme, a real number random generator randomly generates polygon edges (edge lengths and directional angles). The newly generated edge is concatenated with the last edge. To satisfy the simplicity constraint for the polygon (no intersection between edges), each newly generated edge must be checked to ensure that there is no intersection between this edge and any already existing edges. If there is intersection, this edge will be discarded, and another edge will be re-generated and tested. Of course the last edge cannot be randomly generated and is determined by closing up the polygon.

5.2.3 Crossover of Polygons

As described in section 2.6, crossover is the main operation used for reproduction. It combines portions of two parents to create two new individuals, called *offspring*, which inherit a combination of the features of the parents. In general, to design a sensible crossover method for a specific problem is not an easy task. Although crossover operation is applied to the genetic representation (genotypes, strings) of the physical solutions, the crossover scheme must be carefully designed so that the mechanism of the crossover makes sense on the physical solutions (phenotypes), and the crossover offspring inherit features from both parents. Considering this, when designing crossover schemes for polygons, efforts should focus on developing a scheme to inherit and combine features from two parent polygons, although the final implementation is on the genetic representation, which is simply two real strings.

Real Number Crossover

First consider which real crossover operator to use. As introduced in section D.2, there are several common crossover operators for real-coded strings. Here the blend crossover BLX-α [33] is used. BLX-α crossover produces an offspring by picking a value for each gene uniformly from the range formed by the values of two corresponding parent genes with some extension on each side. If the distance between the two parent genes is I, the length of the extension on each side is αI, and α is a user-specified parameter. The advantages of BLX-α are described in section 2.6. For crossover of polygons, BLX-α scheme can be applied to all the paired elements in both *Distance String* pair and *Angle String* pair.

Geometric Constraints

Now consider what must be modified in the blend crossover BLX-α for real-coded strings for polygon crossovers.

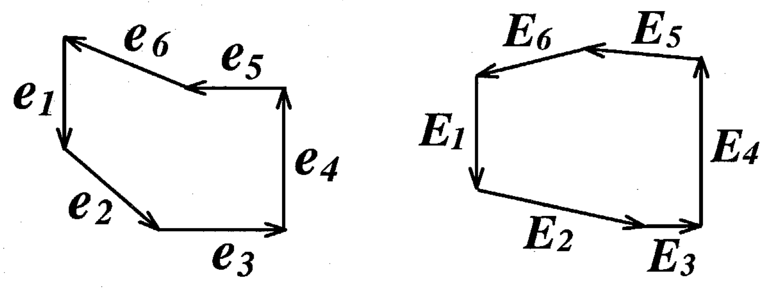
There are two aspects that need to be handled with special care for polygon crossovers. First, each polygon is represented by two real strings, a distance string, and an angle string. For two parent polygons, a pair of distance strings and a pair of angle strings are to be combined. After every pair of genes (real numbers) in these two string pairs is crossed using BLX-α, a new distance string and a new angle string will be generated. But, unfortunately, they are not guaranteed to represent a simple closed 2-D polygon because the blend crossover is randomly carried out, and there may be intersection between edges and the polygon may not be closed. In order to make sure the offspring from the crossover is a simple closed polygon, care must be taken to satisfy the geometric constraints.

To ensure the offspring polygon to be closed, one edge pair must not participate the crossover, and for the offspring this edge will be determined by closing up the polygon. Which pair of edge should be chosen not to participate in the crossover? Remember the purpose of crossover is to generate offspring inheriting common features from the parents. If a pair of edges from two parents are very similar, this feature should be expected to be preserved to the next generation, instead of disrupted. If a pair of edges from two parents are very dissimilar, it should be expected that crossover will explore other possibilities for this edge. This is essentially how the crossover operator balances the effort of exploitation of promising area and exploration of unknown area in the solution space. Under this guidance, the blend crossover should be applied to the edge pairs which are common features in both parent polygons, and the worst matched pair of edges should be chosen as the pair of edges that don't participate the crossover. This way the common features from both parents will be preserved by the crossover, and new features will be introduced by closing up the polygon to replace the worst matched pair of edges. Furthermore, since the angle string pair and distance string pair are crossovered separately, the worst matched pair of genes for each string pair is selected, and they don't necessarily correspond to the same edge pair.

This scheme is shown in Figure 5.3. The crossover is performed for angle strings and distance strings separately. In Figure 5.3, two parent polyons have 6 sides, and angle strings and distance strings with gene length of 6 are paired up. For each string pair, the worst matched pair of genes is chosen first. For the angle string pair, the worst matched pair of genes is gene 6, and for the distance

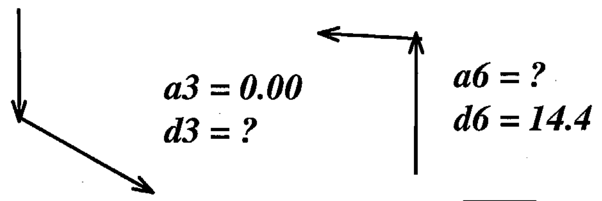
Parent Polygons:

Polygon P_1 ***Polygon P_2***



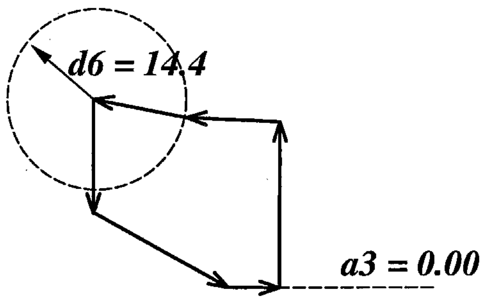
Angle String Pair	4.71	5.41	0.00	1.57	3.14	$\boxed{2.55}$	***P_1***
	4.71	5.90	0.00	1.57	3.05	$\boxed{3.31}$	***P_2***
Distance String Pair	17.4	17.0	$\boxed{15.2}$	21.7	13.0	15.7	***P_1***
	17.8	24.0	$\boxed{4.5}$	26.7	13.5	13.5	***P_2***

Child:



Angle String	4.71	5.67	0.00	1.57	3.10	$\boxed{a6}$
Distance String	17.7	21.6	$\boxed{d3}$	22.3	13.1	14.4

Reconstruction



$d3 = 7.55, \quad a6 = 2.56$

Figure 5.3: Crossover of two polygons

string pair, gene 3. For the angle string pair, every gene pair is crossovered using blend crossover BLX-α except gene 6. For the distance string, every gene pair is crossovered except gene 3. After the crossover, one new angle string and one new distance string are generated, with gene 6 on angle string, a_6, and gene 3 on distance string, d_3, undetermined. These two undetermined values can be determined by the geometric constraint that the two strings should represent a closed 2-D polygon, as shown in the following.

For a closed 2-D polygon with side number of n, the edges are $\vec{e_i}, i = 1, ..., n$. The fact that the polygon is closed can be expressed mathematically as:

$$\vec{e_1} + \vec{e_2} + \vec{e_3} + ... + \vec{e_n} = 0 \tag{5.1}$$

or, in Cartesian coordinates,

$$\sum_{i=1}^{n} d_i \cos a_i = 0 \tag{5.2}$$

$$\sum_{i=1}^{n} d_i \sin a_i = 0 \tag{5.3}$$

where d_i and a_i are the edge length and directional angle of edge $\vec{e_i}$.

For the example in Figure 5.3, a_6 and d_3 can be easily calculated from Equation (5.2) and Equation (5.3): $a_6 = 2.56$, and $d_3 = 7.55$. This procedure to determine the two unknown values of edge length and angle can be shown visually as a geometric reconstruction: by constructing pieces of edges together into a closed polygon, the two unknown values can be geometrically determined. This procedure is illustrated at the bottom of Figure 5.3.

Finally the offspring polygon will be checked to see there are edge intersections. If any, this offspring will be discarded, and the crossover will be performed again.

Polygon Edge Alignment

Now consider another aspect of crossover for polygons. In the section above, crossover is described being carried out on each gene pair. During the procedure, it is assumed that for two similar polygons, the strings representing the polygons are ordered in such a way that similar edges from the two polygons appear on the same gene location on the strings. But this is not necessarily the case. Recall the encoding scheme used to represent polygons: each edge is represented by two real numbers, one on the angle string, one on the distance string. The first edge is randomly chosen and it makes no difference if a different edge is chosen as the first one, as long as the counterclockwise winding sequence is kept unchanged. If $[d_1, d_2, d_3, d_4, d_5]$ and $[a_1, a_2, a_3, a_4, a_5]$ are the distance string and angle string for a 5-side polygon, then strings $[d_4, d_5, d_1, d_2, d_3]$ and $[a_4, a_5, a_1, a_2, a_3]$ represent the same polygon. To ensure that the common features (similar edges) of both parent polygons are on the same gene location on the strings so that similar edges from both polygons will be paired up in the crossover, special care must be taken to achieve the *alignment* of the polygon edges.

Recall the shape matching algorithm illustrated in Chapter 4, the minimum of $L2$ distance function over the moving reference point of one polygon while keeping the reference point of the other polygon fixed, is taken as the shape mismatch. Geometrically what happens is this: choose one vertex from each polygon as the starting vertex (reference point), calculate the $L2$ distance function, and try all the combinations for the starting vertex (mn cases for an m-sided polygon and an n-sided polygon), and the minimum is taken as the shape mismatch. In that sense, polygon edge alignment can be achieved using exactly the same algorithm. The pair of starting vertices corresponding to the minimum of the distance function will be taken as the starting vertices for both polygons in the string representations, and the edge correspondence (alignment) between the two polygons is therefore established.

After the alignment adjustment, the common features on the two parents are paired up, and further crossover will always exploit the maximum level of common geometric features existing between the two parent polygons.

5.2.4 Crossover of Polygons with Different Side Numbers

The crossover scheme described above has a limitation: it only works on two polygons with the same number of sides. The mechanism of the crossover scheme is based on the pairing up of corresponding edges in two polygons, and each edge in one polygon must have a corresponding edge in the other polygon to crossover with. Using such a crossover scheme restricts the mask-layout synthesis to a subspace of polygons that all have a specified number of sides. The appropriate number of sides for the mask-layout synthesis must be guessed in advance, and only a subspace of the whole polygon space, polygons with this pre-specified number of sides, is explored. The optimal mask-layout may not be found simply because it lies outside of this subspace. A variable gene length crossover scheme for polygons was developed so that polygons with different numbers of sides can be explored by the algorithm, and the chance to find the optimum is greatly increased.

To design a variable gene length crossover scheme for polygons, it was noticed that a polygon's shape is unchanged after adding some vertices onto the edges of the polygon. In other words, although an n-side polygon normally is viewed as having n edges, it can also be viewed as having more than n edges because one edge can be broken into two edges with same directional angle. In this sense, if two polygons with n sides and m sides respectively, where $m > n$, are considered, both polygons can be viewed as m-sided by adding $m - n$ vertices onto the edges of the n-sided polygon. After that, these two m-sided polygons can be crossovered using the scheme introduced previously. The offspring from such a crossover will be an m-sided polygon. Some vertices of the offspring polygon (usually ones connecting the most collinear edges) are then removed to produce the final polygon with number of sides which is generated randomly between n and m.

Having introduced an approach (called an *add-then-remove scheme*) to the crossover problem of polygons with different numbers of sides, where to add the new vertices onto the edges of a polygon is still a challenging task. An example is used below to illustrate the scheme used here to determine where to add new vertices.

Figure 5.4 shows the add-then-remove scheme introduced above. Two polygons to be crossovered are shown at the top of the figure: one with 4 sides, the other with 6 sides.

First, a scheme to determine where to put the new vertices on the polygon with fewer sides is needed. The locations of new vertices should be chosen in such a way that after insertion of new vertices and crossover, the offspring polygon combines common features of both parent polygons. In other words, the insertion of new vertices should not break the already established common feature (edge) correspondence, and new edge correspondence should also be established by the insertion. First, polygon edge alignment is conducted to determine the first edge for both polygons, and get the distance string and angle string representation for both polygons. To determine the insertion positions, two parent polygons are scaled to same perimeter length (1.0 in the example). Using a line segment from 0.0 to 1.0 to represent the perimeter of a polygon, each vertex of the polygon corresponds to a point on the line segment. For the square in Figure 5.4, the vertices are at 0.25, 0.5, 0.75, and 1.0, and for the hexagon, the vertices are at 0.2, 0.35, 0.5, 0.7, 0.85, and 1.0.

In order to determine where to put the new vertices on the polygon with fewer sides (the square), it must be established which pair of vertices on both polygons correspond to each other. For each vertex on the polygon with fewer sides, the vertex closest to it on the line segment of the other polygon (the polygon with more sides) is chosen to be the corresponding vertex. In the example shown in Figure 5.4, for vertex v_1 of the square, the vertex V_1 of the hexagon is the corresponding vertex. Vertex v_2 corresponds to vertex V_3; vertex v_3 to vertex V_4, and vertex v_4 to vertex V_6. The two

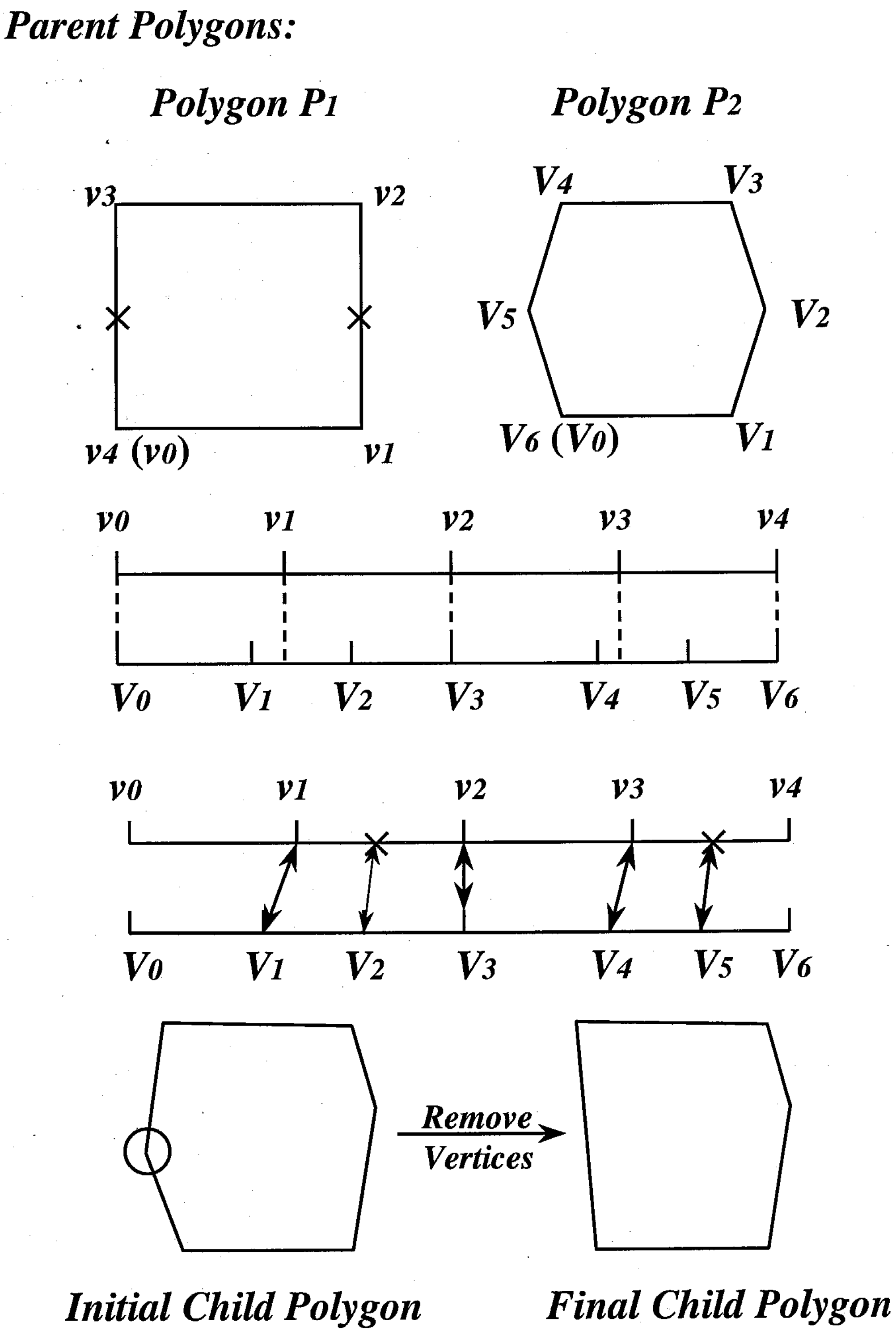


Figure 5.4: Add-then-remove scheme for crossover of polygons with different numbers of sides

segments of the two polygons with corresponding vertices as end points are called corresponding segments. For example, the segment from v_1 to v_2 of the square and the segment from V_1 to V_3 of the hexagon are corresponding segments. Now two vertices are left uncorresponded on the hexagon, vertex V_2, and vertex V_5. Vertex V_2 is on segment V_1 to V_3 which has corresponding segment v_1 to v_2 on the square, and a new vertex is generated on this corresponding segment of the square with its position relative to this corresponding segment exactly the same as vertex V_2 is relative to the corresponding segment on the hexagon. The corresponding vertex of vertex V_5 is generated the same way. The "×" symbols in Figure 5.4 show where the new vertices are inserted.

After insertion of the new vertices, the two polygons (now with the same number of sides) are crossovered to generate the initial child polygon, and then a number between the side numbers of the two parents is randomly generated as the side number of the final child polygon, and some vertices of the initial child polygon (usually the most collinear vertices) are then removed to produce the final polygon with the specified number of sides. In the example, the parents have side numbers of 4 and 6, the initial child polygon has 6 sides. The randomly generated final polygon side number is 5, and one vertex of the child polygon (the one in the circle) is removed to produce the final polygon with 5 sides.

5.2.5 Mask Mutation

Although in general crossover is the driving force for the convergence of a Genetic Algorithm, without mutation the convergence may be trapped in local optimum. As described in section 2.7, mutation enables new features to be introduced into a population so that premature convergence can be avoided. Mutation makes the whole solution space reachable for every generation during the evolution, and it provides a chance for the algorithm to explore areas that have been lost through the crossover process of the previous generations.

For the problem here, the purpose of the mutation operation is to introduce new polygon edge features into the current generation. Mutation is applied on each individual, polygon in the case here. Using the edge-coding scheme, mutation should be applied to the distance string and the angle string separately. One gene in each string is replaced with a randomly generated value in some range. Geometrically, that means the edge length of one edge ($\vec{e_1}$) and the angle of another edge ($\vec{e_2}$) are changed. Similar to the crossover case, to satisfy the closeness constraint of the polygon, the angle of $\vec{e_1}$ and the edge length of $\vec{e_2}$ cannot be kept unchanged and should be determined according to Equation (5.2) and Equation (5.3). Figure 5.5 illustrates this mutation scheme and the polygon reconstruction.

The mutation scheme introduced above changes the shape of the polygon without changing the number of sides of the polygon. For the Genetic Algorithm to work on polygons with different side numbers, a mutation operation for changing the polygon side number is desired. As the evolution proceeds, the crossover operation will make the side numbers of polygons in one generation converge. At some point, all the polygons in one generation will have the same number of sides. Without a mutation operation to change the side number, further evolution will be working on only a subspace of polygons with a fixed side number. If it turns out the optimum solution has a larger side number and therefore lies out of this subspace, the algorithm will never reach this optimum. A mutation operation to change the polygon side number and therefore introduce candidates with different topology is needed for the Genetic Algorithm to avoid premature convergence.

The implementation of such mutation is simple. To add edges into a polygon, vertices can be inserted along the polygon perimeter. This doesn't change the geometry of the polygon, but the string length is changed. Also vertices can be removed from a polygon to decrease the polygon side number. Since the purpose of this mutation is not to change the shape of the polygon, when removing vertices from a polygon, the vertices connecting the most collinear edges or very short edges will be removed first. This way the polygon shape will not be severely changed.

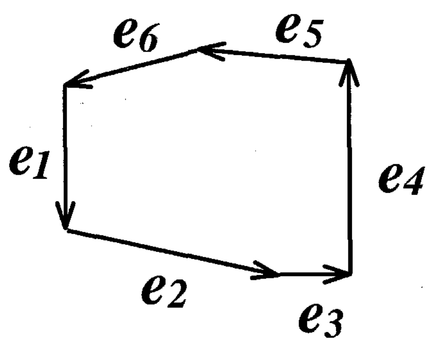
5.3 Selection Scheme

Although genetic operations such as crossover and mutation are the means for a Genetic Algorithm to explore and exploit the solution space searching for the optimum solution, the principle of evolution, survival of the fittest, is achieved through a well-designed selection pressure. An appropriately designed selection scheme should apply proper selection pressure to individuals in a population, so that the better-performing individuals will have a higher chance to survive the competition and reproduce, while worse-performance individuals are not discarded totally without a chance.

As described in section 2.5, a selection scheme assigns a selection probability for each individual in a population. Parents will be selected according to the assigned probability to go through genetic operations such as crossover and mutation to generate offspring. A linear ranking selection algorithm is used for the problem here because of its advantages (see section 2.5.1 for details).

For a linear ranking selection, the ranking of all chromosomes in a population (instead of the raw fitness values) is used to assign the selection probability. An algorithm will map the rank of

Initial Polygon:

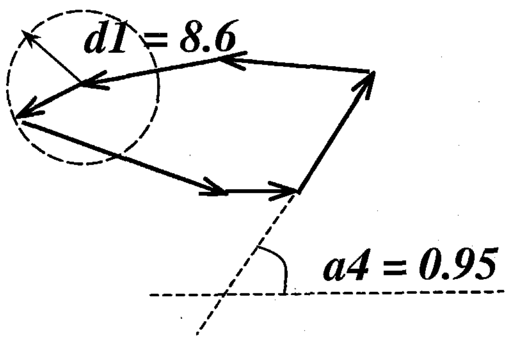


Angle String	4.71	5.90	0.00	1.57	3.05	3.31
Distance String	17.8	24.0	4.5	26.7	13.5	13.5

Mutation: $a1 = ?$, $d1 = 8.6$ $a4 = 0.95$, $d4 = ?$

Angle String	$\boxed{a1}$	5.90	0.00	$\boxed{0.95}$	3.05	3.31
Distance String	$\boxed{8.6}$	24.0	4.5	$\boxed{d4}$	13.5	13.5

Reconstruction



$a1 = 3.35, \quad d4 = 14.45$

Figure 5.5: Mutation of a polygon

a chromosome in a population into a *reproductive fitness value* for this chromosome, and then the reproductive fitness values will be used in a fitness proportionate selection scheme to calculate the selection probability for each chromosome. Let F'_k be the reproductive fitness value for the kth chromosome in the ranking of a population; the linear ranking takes the following form:

$$F'_k = q - (k-1) \times \frac{q - q_0}{pop\_size - 1}$$

where parameter q is the reproductive fitness for the best chromosome, and q_0 is the reproductive fitness for the worst chromosome. Intermediate chromosomes' fitness values are decreased from q to q_0 linearly. When q_0 is set to be 0, it provides the maximum selective pressure.

Note that

$$F'_k = q(1 - (k-1) \times \frac{1 - \frac{q_0}{q}}{pop\_size - 1})$$

Since q is a common factor, its value is not important because for fitness proportionate selection, only the relative fitness matters. Let $q = 1$, then

$$F'_k = 1 - b \times \frac{k-1}{pop\_size - 1}$$

where $b = 1 - q_0$ is a value between 0 and 1. The value of parameter b can be used to control the selection pressure and is referred to as *selection bias value.* When $b = 0$, the reproductive fitness values for all chromosomes are all the same, and therefore there is no selection pressure at all. When $b = 1$, $F'_k = 0$ for the worst chromosome ($k = pop\_size$), and it will never be selected. Figure 5.6 illustrates the linear ranking selection scheme.

In this application, the selection bias value b is gradually increased as the genetic iteration proceeds to control the rate of convergence. During the initial stage the selection pressure is light so that more areas in the solution space can be explored and premature convergence can be prevented, and during the final stage the selection pressure is heavy so that the population will effectively converge to the final solution.

Recall that parent selection is only one of the two kinds of selection in one iteration of a Genetic Algorithm. After the offspring are generated by crossover and mutation, a generational selection is performed to select individuals from both the parent population and the offspring population to create the next generation. The elitist selection described in section 2.5.2 is used here to select

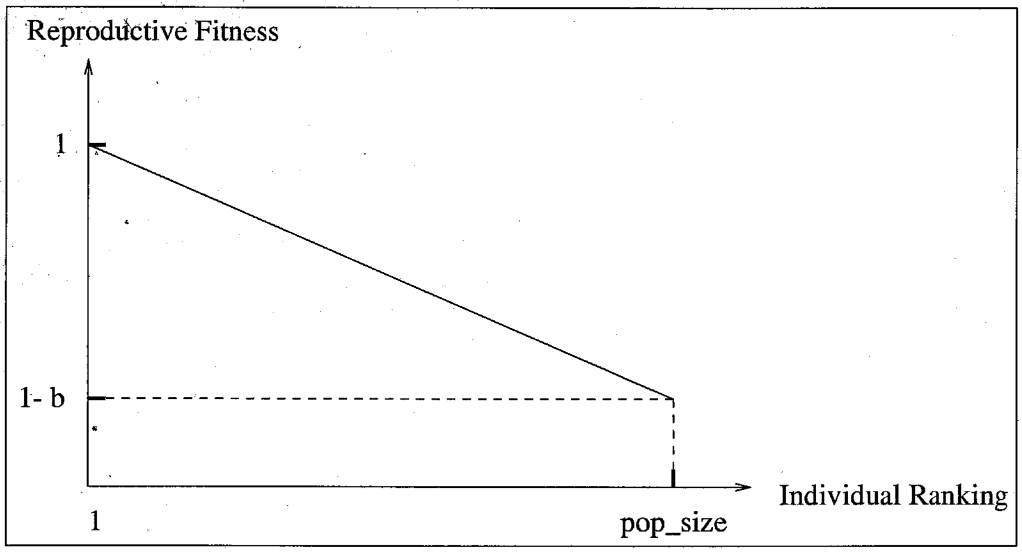


Figure 5.6: Selection pressure controlled by selection bias b

the best chromosomes from both the parents and the offspring to create the next generation while keeping the population size unchanged. This way the best-performance individuals will always be preserved along the evolution.

5.4 Symmetry: Computational Considerations

Because of the geometric constraint requiring simple closed polygons, not all polygons generated are valid. The polygons need to be checked to guarantee their validity before they are evaluated and participate in further evolution. For a randomly generated polygon or an offspring polygon from crossover or mutation, the polygon generation mechanism described above produce the result that the chance of a polygon with more sides to be invalid is higher than a polygon with fewer sides. Therefore more computational time is needed to generate valid polygons with more sides. Also polygons with more sides need more time for crossover and mutation because more sides means longer gene string length. Considering these factors, computational costs can be significantly reduced in some special cases when polygons have symmetries and can be represented by strings with length less than the number of sides in the polygon.

Because of the physical crystalline structure of silicon, the anisotropic etchants' etch rates exhibit symmetry. The etch rate for the direction indicated by the Miller indices (l, m, n) will be the same no matter where this direction is physically. For a (100) wafer, the SEGS etching simulator assumes the x axis and y axis are constructed along two perpendicular <110> directions, and the etch rate is symmetric to both x axis and y axis. If a mask-layout is quadri-symmetric (symmetric to both x axis and y axis), then the etched device geometry will also be quadri-symmetric. Conversely, if a target device shape is quadri-symmetric, the search effort can be constrained to search for quadri-symmetric mask-layouts.

If a polygon is quadri-symmetric, then only one-fourth of the edge features are needed to represent the polygon. Using the edge coding scheme, strings with length of one-fourth of the polygon side number are enough to encode the polygon. In more detail, only one-fourth portion of the edges, in the first quadrant, is recorded in the distance string and angle string. Some slight modification needs to be performed for the polygon initialization and the genetic operations, such that they are only performed in the first quadrant. Before evaluation the mask, the encoded first quadrant portion of the polygon is reflected about x axis and y axis to reconstruct the whole polygon. By utilizing the quadri-symmetry of the target shape and mask-layouts, the computational time for validity checking and genetic operations is largely reduced. In the following chapters, quadri-symmetric mask-layouts will be searched for quadri-symmetric target shapes.

5.5 Test: Searching for a Quadri-Symmetric Polygon

To test the performance of the Genetic Algorithm structure just constructed, experiments are conducted to search for a quadri-symmetric polygon. The etching simulation is not plugged in here since only the performance of the searching mechanism is tested, *e.g.*, coding, initialization, crossover, mutation, and selection. A test to search for a pre-specified polygon is enough to serve this purpose.

A quadri-symmetric polygon (a cross as shown in Figure 5.7) is given as the target shape. The Genetic Algorithm procedure is performed to find a polygon close to the target polygon. The shape matching algorithm evaluates the performance of each candidate polygon. The synthesis run was conducted on a Sun Ultra10 workstation with a 440 MHz CPU clock speed, and it took about 2 minutes. The parameter settings for the GA are shown in Table 5.1. For each generation, each candidate polygon is compared with the target polygon using the shape matching algorithm and the shape mismatch values are recorded for each candidate polygon. The average shape mismatch (both shape mismatch and size mismatch) is calculated for each generation, and Figure 5.8 shows the convergence of the average shape mismatch along with the iterations.

The geometries of the best resulting polygons in iteration 1, 5, 10, 15, 24, and 50 are shown in Figure 5.9, and the convergence of the polygon shapes to the target shape can be easily observed.

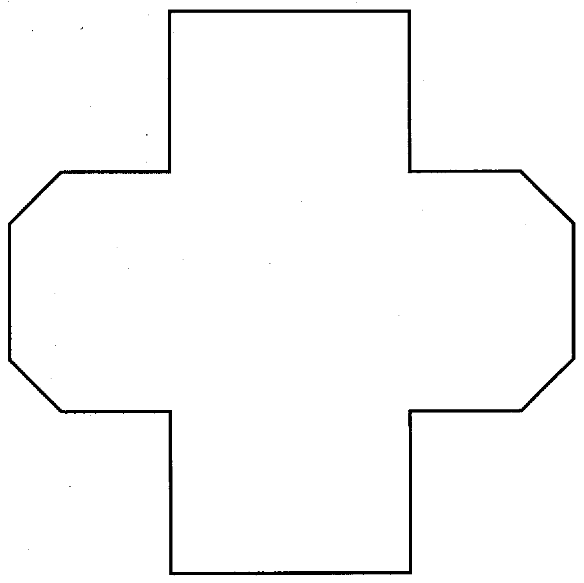


Figure 5.7: A cross as the target shape

Table 5.1: GA parameter settings

GA Parameter	Value
Iteration Number	50
Population Size	50
Min Polygon Side Number	4
Max Polygon Side Number	28

Table 5.2: Synthesis data showing convergence

Iteration	Side Number	Shape Mismatch	Size Mismatch
1	20	0.440	0.416
5	12	0.235	0.118
10	12	0.212	0.048
15	16	0.146	0.089
24	16	0.119	0.086
50	16	0.023	0.022

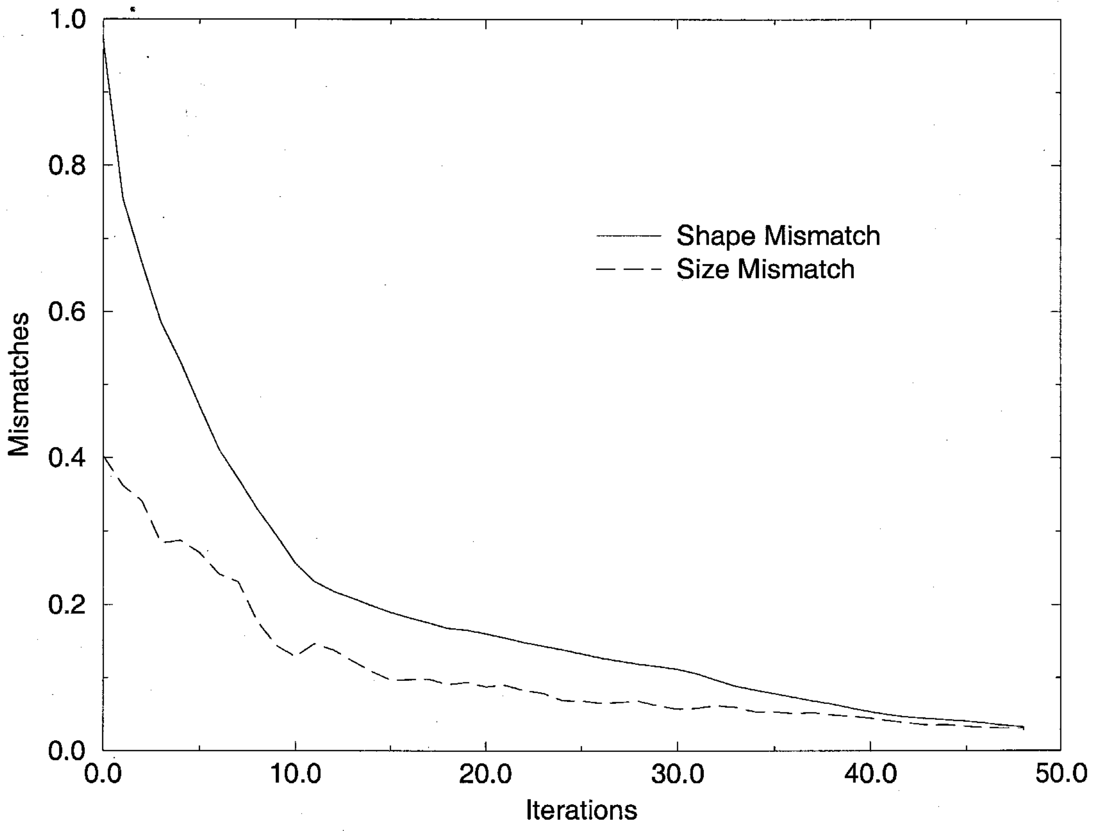


Figure 5.8: Convergence curves of shape and size mismatches

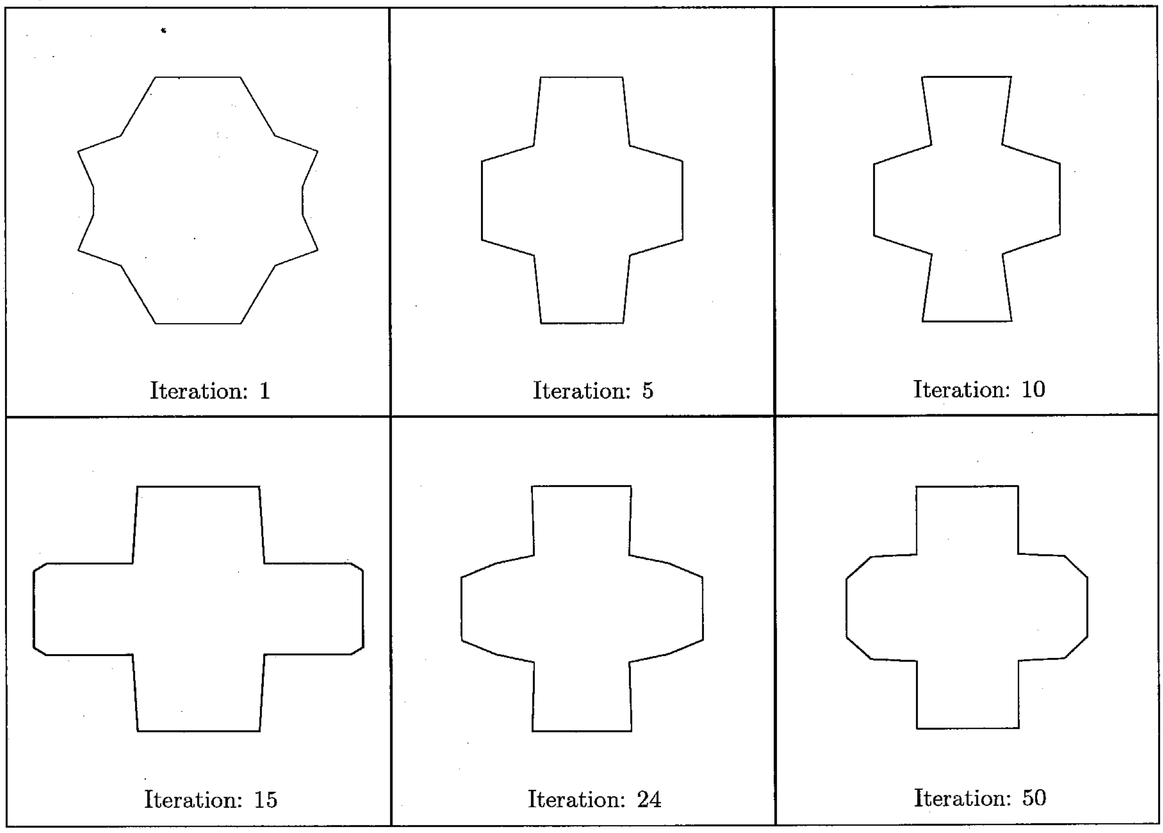


Figure 5.9: Best polygons at different iterations

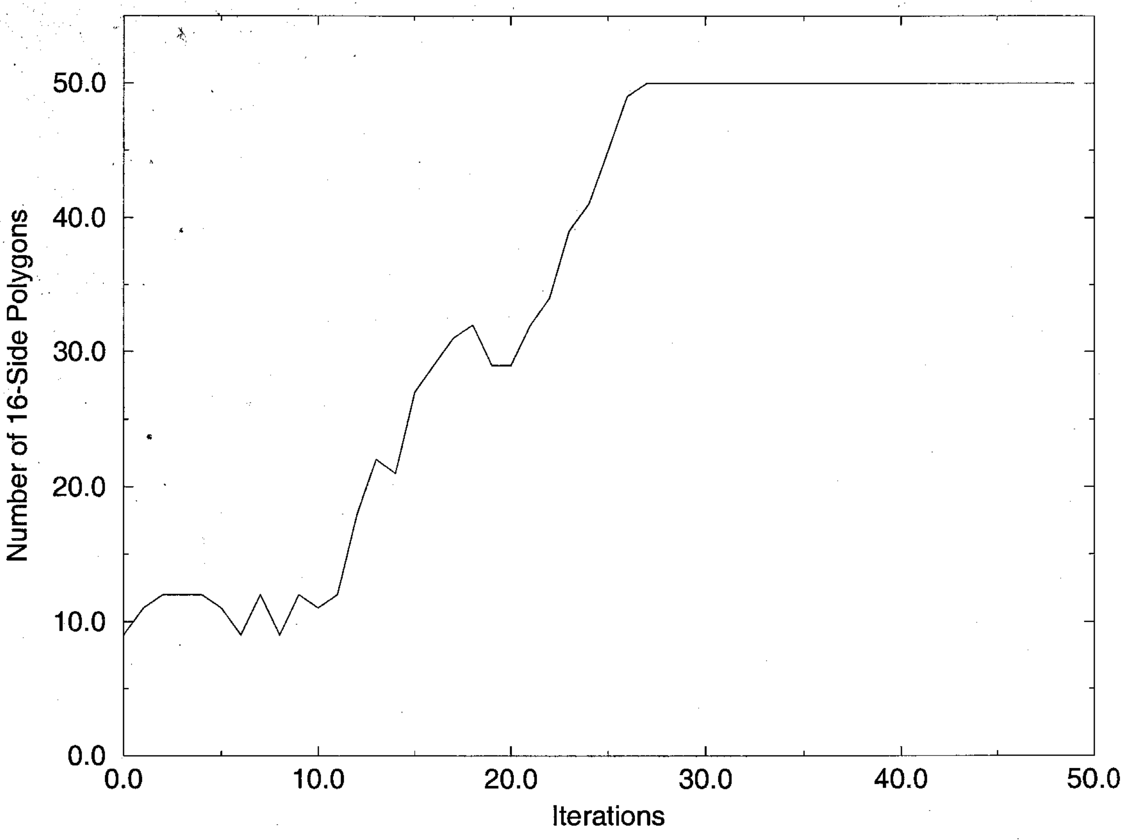


Figure 5.10: Convergence of the number of polygons with 16 sides to the population size

Detailed performance information about these six polygons is shown in Table 5.2. The shape and size mismatch values between the best polygon and the target polygon for each iteration are getting smaller as the synthesis proceeds. At iteration 50, the best resulting polygon is close to the target polygon and it's taken as the optimum and the synthesis is stopped.

The target polygon has 16 sides. As shown in Table 5.1, during the synthesis, polygons with number of sides between 4 and 28 are explored. A well designed GA should gradually shift the searching effort to the subspace of polygons with 16 sides, and the final result should have 16 sides. The number of the candidate polygons with 16 sides in each generation is recorded, and Figure 5.10 shows the convergence of the number of the candidate polygons with 16 sides to the population size (50) as the synthesis proceeds.

To demonstrate the importance of having a variable gene length GA structure to evolve polygons with proper side number, a synthesis experiment is performed on polygons with a fixed side number. For the same target polygon (shown on the left side of Figure 5.11) with 16 sides, if the GA tries to find it in the subspace of polygons with 12 sides, the synthesis will produce the polygon shown

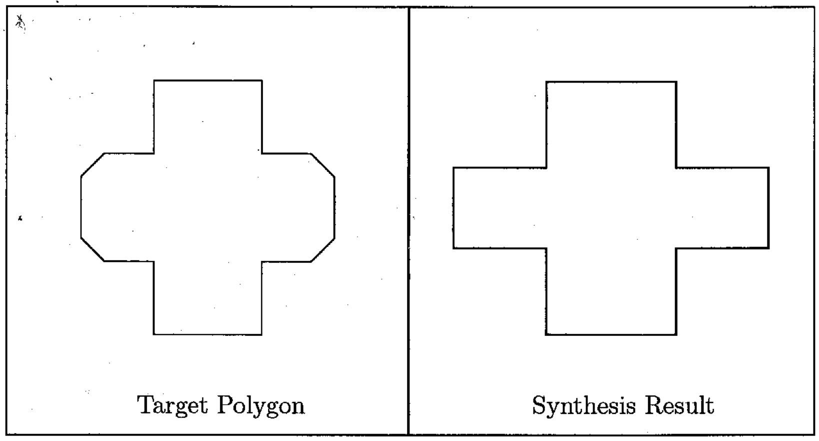


Figure 5.11: Target polygon and the search result using fixed polygon side number GA

on the right side of Figure 5.11 as the best result. Because of the difference between polygons with different numbers of sides, it's obvious that a polygon with 12 sides cannot be found having strong resemblance to a 16-side polygon. If the search is conducted in a subspace with side number more than 16, polygons close to the target 16-side polygon can be found, but more computational time is needed because of the redundant edges.

5.6 Process Flows

The preceding sections illustrate the coding scheme and genetic operations of the Genetic Algorithm for mask-layouts. To synthesize process flow, the process flow parameters also need to be encoded and manipulated by genetic operations so that a search for the optimum process flow can be performed. To demonstrate how to construct the coding scheme and genetic operations for process flow parameters, this section describes the implementation details of the coding scheme and genetic operations for a two-step wet etching process.

In a two-step wet etching, two different etchants are applied in sequence onto a mask-layout, each for some etch duration time, to generate the final device geometry. The design parameters for the process flow are the etchant sequence and the etch durations:

etchant sequence:	e_1, e_2
etch durations:	t_1, t_2

where e_1 and e_2 are the index numbers for the first etchant and the second etchant, and t_1 and t_2 are the etch times for the first and second etch step.

Etchant Sequence

Let us first consider the etchant sequence. Suppose m different etchants are available to choose from for the two-step wet etching process, and they are identified as $\#1, \#2, ..., \#m$; therefore, the possible values of e_1 and e_2 are integers between 1 and m, *i.e.*,

$$e_1,\ e_2 \in \{1, 2, ..., m\}$$

Also since the values of e_1 and e_2 are unrelated to each other, they can be evolved independently.

Etch Durations

For the etch durations, as shown below, the values of t_1 and t_2 actually are related, and also the range of possible values for t_1 and t_2 is not fixed for different etchants. A single variable will be constructed as the etch time parameter to participate in the evolution, and the values of t_1 and t_2 will be determined accordingly.

For the synthesis problem, given a target shape, the total etch depth is known. For a two-step etching process, the total etch depth is decided by the etch times and the <100> etch rates for both etchants as follows:

$$\text{Total Etch Depth} = t_1 \times r_1 + t_2 \times r_2 \tag{5.4}$$

where r_1 and r_2 are the <100> etch rate for the first and second etchant. Therefore,

$$t_2 = \frac{\text{Total Etch Depth} - t_1 \times r_1}{r_2}$$

In other words, t_2 can be calculated once t_1 is known and the etchant sequence is known (therefore the etch rates r_1 and r_2 are known). Therefore, only t_1 needs to participate in the Genetic Algorithm evolution.

Now consider the range of possible values for etch time t_1. Obviously, from Equation (5.4),

$$t_1 \in [0, \frac{\text{Total Etch Depth}}{r_1}]$$

But since the <100> etch rates (r_1) can be different for different etchants, the range for possible values for etch time t_1 will be different for different etchants. To overcome this problem, a new variable T_1 is constructed so that

$$T_1 = t_1 \times r_1$$

The range of possible values for variable T_1 is

$$T_1 \in [0, \text{Total Etch Depth}]$$

T_1 is the etch rate variable that participates in the GA evolution, and for a specific value of T_1, the values of t_1 and t_2 can be calculated as

$$\begin{aligned} t_1 &= \frac{T_1}{r_1} \\ t_2 &= \frac{\text{Total Etch Depth} - T_1}{r_2} \end{aligned}$$

Summary

In summary, the design parameters for the process flow are as follows:

1. etchant sequence e_1, e_2, integers, with range of $[1, m]$.

2. etch time variable T_1, real number, with range of $[0, \text{Total Etch Depth}]$.

A design point with $e_1 = a$, $e_2 = b$, and $T_1 = c$ means the first etchant is etchant $\#a$, the second etchant is etchant $\#b$, the first etch time is $\frac{c}{r_a}$, and the second etch time is $\frac{\text{Total Etch Depth}-c}{r_b}$.

For the fabrication process coding scheme, two integers and one real number are used to encode e_1, e_2 and T_1. The initialization will generate the initial population by randomly sampling in the corresponding solution spaces ($[1, m]$, $[1, m]$, and $[0, \text{Total Etch Depth}]$). The genetic operation (crossover, mutation) schemes for real coding (see Chapter 2 for detail) can be applied. For e_1 and e_2, round-off needs to be performed to get integer values.

5.7 Summary

This chapter introduced in detail the structure and key components of the Genetic Algorithm approach to the mask-layout and process synthesis problem. For mask-layouts, coding scheme, crossover, and mutation mechanisms have been constructed to represent and manipulate the evolution of polygons with variable numbers of sides. Special care needs to be taken to satisfy the geometric constraint of simple closed polygons. Linear ranking selection and elitist selection are used to select candidate solutions to reproduce offspring and create the next generation. The Genetic Algorithm structure for mask-layout synthesis was tested by an example to search for a desired quadri-symmetric polygon, and the synthesis results verified the stable and fast convergence of the algorithm. To synthesize process flows, coding scheme and genetic operations are needed to encode and evolve process parameters. To demonstrate the general procedure for manipulation of process flow parameters, implementation details of the coding scheme and genetic operations were described for a two-step wet etching process. Now that the key components of the Genetic Algorithm approach to the mask-layout and process synthesis problem have been constructed, the next chapter will apply the approach to the synthesis of mask-layout and process for bulk wet etching.

Chapter 6

Mask-layout and Process Synthesis for Bulk Wet Etching

6.1 Overview

In the preceding chapters, the main components of the Genetic Algorithm approach to the mask-layout and process synthesis have been introduced. Chapter 5 described the structure and implementation of the Genetic Algorithm approach to the synthesis problem; Chapter 3 introduced the SEGS wet etching simulator which is used to simulate the etching process; Chapter 4 described the shape comparison algorithm which evaluates the performance of the candidate solutions. In this chapter, all the components will be integrated, and synthesis examples will be shown to demonstrate the capacity of the algorithm, and provide a basis for further extension of the method.

Designing corner compensation structures is one of the most challenging tasks when designing mask-layouts for anisotropic wet etching. Mask-layout synthesis experiments are conducted here on two target shapes, and different compensation patterns are generated by the Genetic Algorithm synthesis method. An additional example concerns two-step wet etching process in which two different etchants will be applied in sequence to generate the final device shape. For a given target shape, synthesis is conducted to search for not only a proper mask-layout, but also the right etchant sequence and the etch durations for both steps. A synthesis involving both the mask-layout and the process flow is more complex, and automatic design tools are helpful to synthesize the right process recipe.

6.2 Corner Compensation

Anisotropic wet etching of (100) silicon wafer is an important technique for silicon micromachining. Rectangular diaphragms can be formed by this technique with little difficulty. But when etching

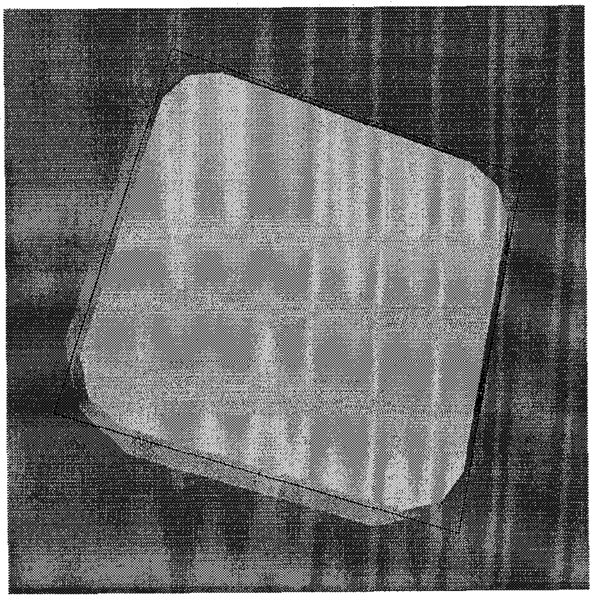


Figure 6.1: Corner undercutting of a square mask in anisotropic etching

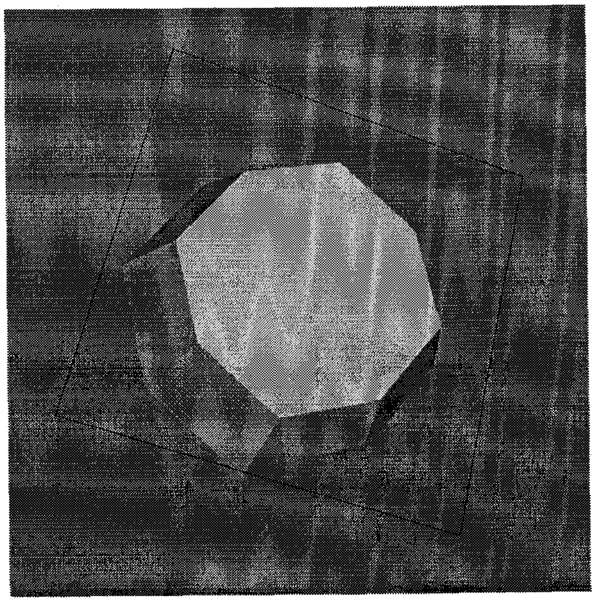


Figure 6.2: Octagonal shape resulted from long etch time

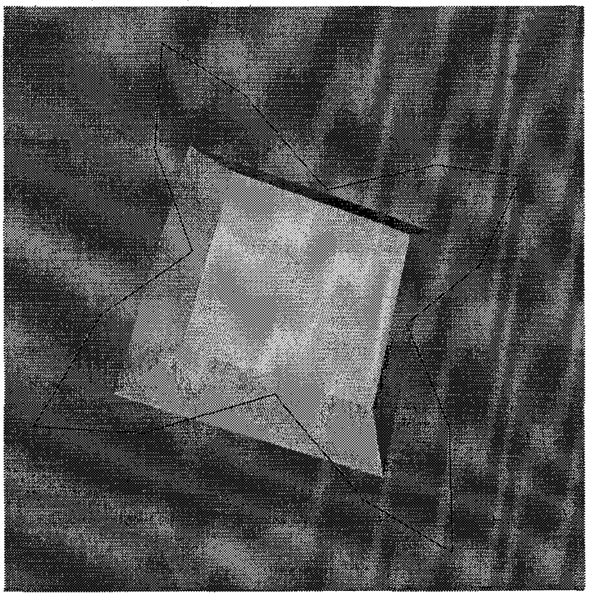


Figure 6.3: A mask with compensation structures and the resulted square mesa

rectangular convex corners in silicon using anisotropic etchants, deformation of the edges always occurs due to undercutting. The corner undercutting starts at the convex corner of the mask and increases gradually. Figure 6.1 shows a typical example of this undercutting phenomenon. The convex corners of the square mask are undercut by the fast moving etching planes, and as a result, two etching fronts appear on each convex corner. If the etching time is long enough, the square mesa eventually becomes octagonal (Figure 6.2). This is an unwanted effect in micromachining silicon, e.g., in the fabrication of mechanical sensors for measuring acceleration, where perfect 90° convex corners are mandatory for good device prediction and specification. The undercutting on convex corners has been one of the obstacles to the implementation of more complicated structures, such as mesas and right angle grooves.

The undercutting of convex corners in anisotropic etching can be reduced or even prevented by adding the so called *compensation structures* to the convex corners on the etching mask. For example, to fabricate a square mesa, instead of using a square mask which has undercutting problem as shown in Figure 6.1 and Figure 6.2, a mask with extra boundary features at the convex corners as shown in Figure 6.3 can be used. The compensation structures withstand extra etching time so that the convex corners will not be attacked. The design of a compensation structure is related to the specific etchant as well as to the structure to be formed, as the nature of undercutting is dependent on the etching conditions (such as the composition of the etchant, temperature, *etc.*). Also, since the undercutting is a function of etch time, the size of the compensation structure is directly related to the desired etch depth. Different compensation structures have been proposed. Abu-Zeid uses square patterns and superimposed compositions of square and rectangular patterns on the convex corners on the mask for compensation purpose [1]. A more straightforward design is to add triangle patterns to the convex corners [94, 120]. Bao *et al.* use additional <110> strips [11] and Mayer *et al.* use a <100> bar on convex corners [76]. The compensation patterns usually need a large area when etching depth is deep. Zhang *et al.* proposed a compensation pattern for restricted area such as narrow groove [123].

Different compensation patterns are illustrated in Figure 6.4. As can be seen in Figure 6.4, most compensation structures are composed of edges in certain particular orientations such as <110>, <100> and <210>. These orientations are chosen because it is relatively simple to analyze the etching of structures composed of edges in these orientations, and for some cases, the relationship between the feature length of the compensation structures and the etching depth can be theoretically established. However, these compensation patterns are not necessarily optimized when considering the performance and the area they need. More complex compensation structures can be synthesized by trying other possibilities, and for a given device shape and surrounding geometry constraints, compensation structures should be specially designed to meet the requirement. In the following sections, for two target shapes, a square mesa and a cross groove, mask-layout synthesis using the Genetic Algorithm approach will be conducted to generate various compensation patterns.

6.3 Square Mesa

6.3.1 Etch Rate Data

The etching characteristics for KOH etchant have been studied by various groups for different concentrations and different temperatures [98, 103, 109, 121]. Literature shows the etch rate in <111> direction is very small, usually less than 1 μm/hr [98, 109], and the etch rate ratios are about 1.9:1 for <110>:<100> and 1.6:1 for <311>:<100> [103, 121]. For the synthesis tests here, etch rates for 45wt.%, 58 °C KOH Solution as shown in Table 6.1 are used. The etch rate in <100> direction is 23 μm/hr.

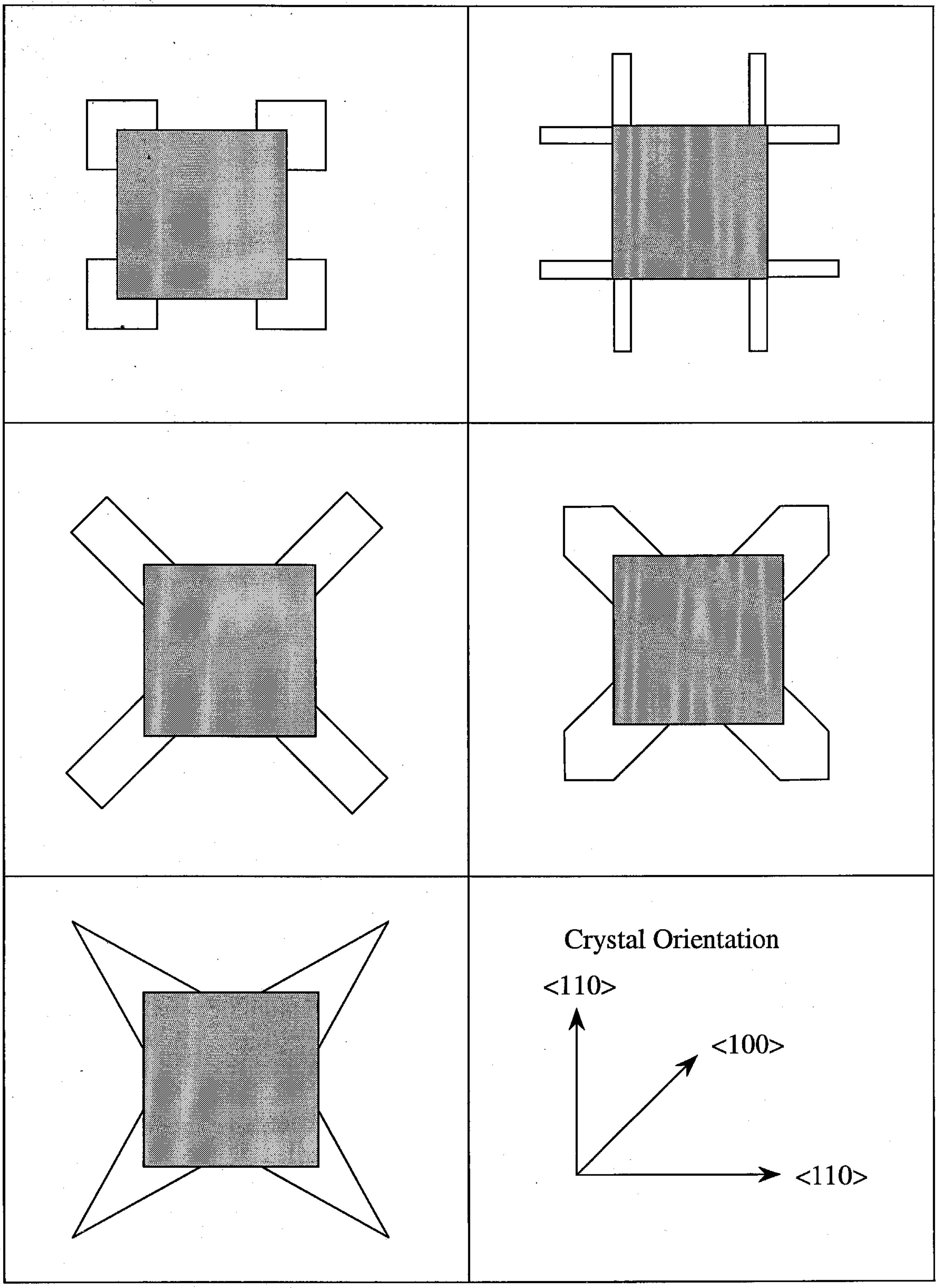


Figure 6.4: Compensation pattern examples

Orientation	Etching Rate (μm/hr)
100	23
110	44
311	37
111	1

Table 6.1: Etching rates for 45 wt.%, 58 °C KOH Solution

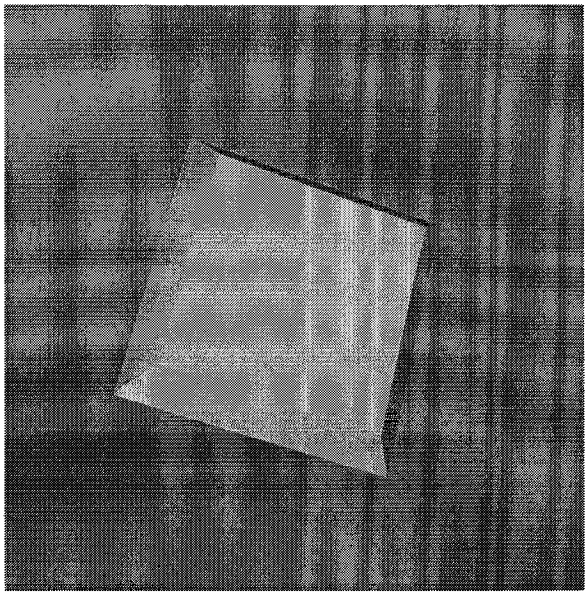


Figure 6.5: A square mesa

Parameter	Value
Population Size	60
Iteration Number	70
Minimum Polygon Side Number	4
Maximum Polygon Side Number	28

Table 6.2: Evolution parameters for GA

6.3.2 Target Shape

The target shape in the first mask-layout synthesis example is a square mesa as shown in Figure 6.5. The sidewalls of the mesa are $\{111\}$ planes whose etching rates are the slowest. For KOH etching, if an edge of a mask is aligned with <110> direction, the etched sidewall will be $\{111\}$ planes since <111> direction has the lowest etch rate. Therefore, for a square mesa generated by KOH etchant, if the top-layer edges of the mesa are aligned with <110> direction, the sidewalls should be $\{111\}$ planes. Because of this, the sidewalls of the target square mesa are set to be $\{111\}$ planes. This way the sidewalls of the target shape and sidewalls of simulated shapes will be in the same direction, and the search will focus on finding compensation structures.

The feature length and etching depth are two parameters of the square mesa. As illustrated in section 6.2, the shape and size of the compensation structures are dependent on the feature length and etching depth of the shape to be formed. Mask-layout synthesis will be conducted for different combination of feature length and etching depth for the square mesa.

6.3.3 Test

As described in Chapter 4, the target shape is actually represented by polygon layers. For the square mesa, five layers are used, and all the vertical intervals are equal to 23 μm. The total etching depth is therefore $23 \times 4 = 92\mu m$. Since the etch rate for <100> direction is 23 μm/hr, the total etching time will be 4 hours and the etching simulation will produce an etching contour for every hour so that the simulated shapes also have five polygon layers with equal distance of 23 μm.

For the first test, the feature length of the square mesa (the top-layer edge length) is 500 μm, and the target shape is shown in Figure 6.6. After some preliminary tuning, the evolution parameters used in the Genetic Algorithm were set as shown in Table 6.2.

The synthesis task was conducted on a Sun Ultra 10 workstation (440 MHz), and it took about

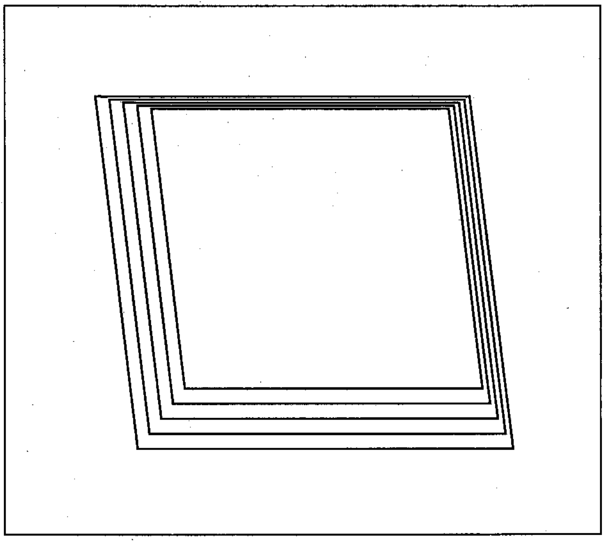


Figure 6.6: A square mesa represented by polygon layers

Table 6.3: Synthesis data showing convergence

Iteration	Side Number	Shape Mismatch	Size Mismatch
1	20	0.166	0.221
5	24	0.187	0.142
10	28	0.123	0.052
21	24	0.019	0.026
50	24	0.013	0.007

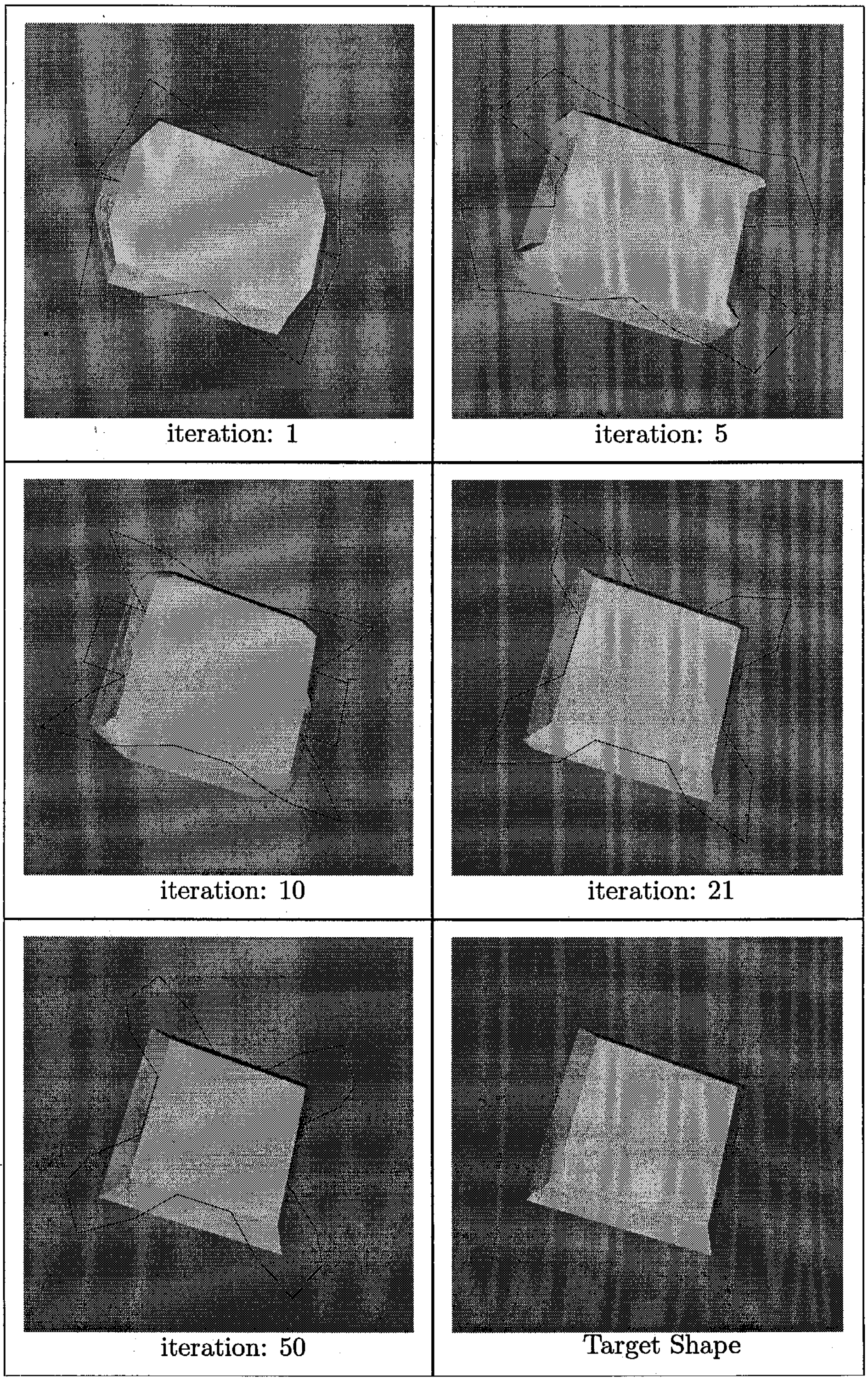


Figure 6.7: Synthesis results: mask-layouts and simulated shapes

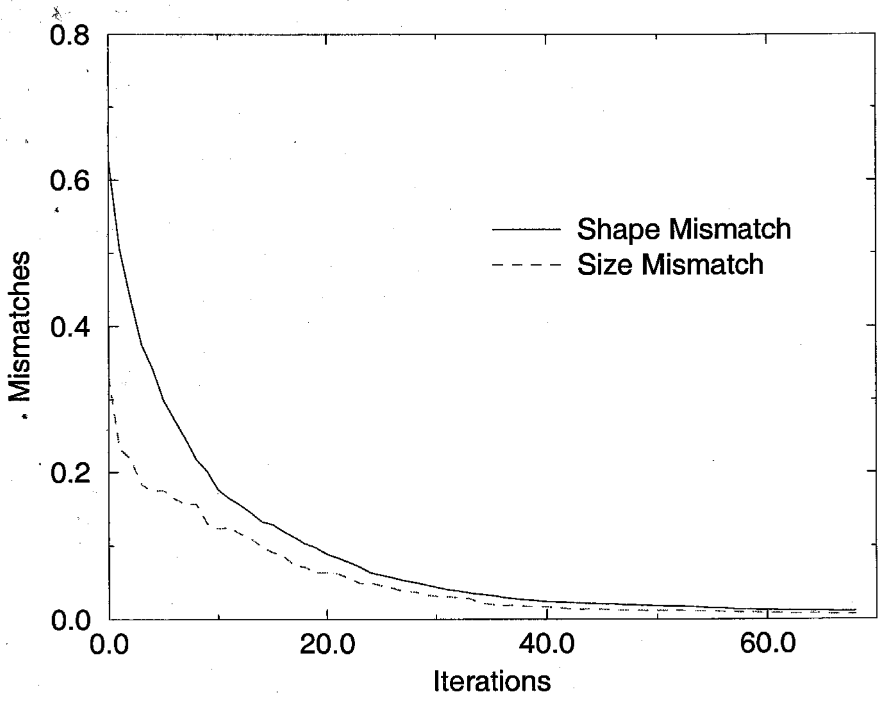


Figure 6.8: Convergence curves of shape and size mismatch values

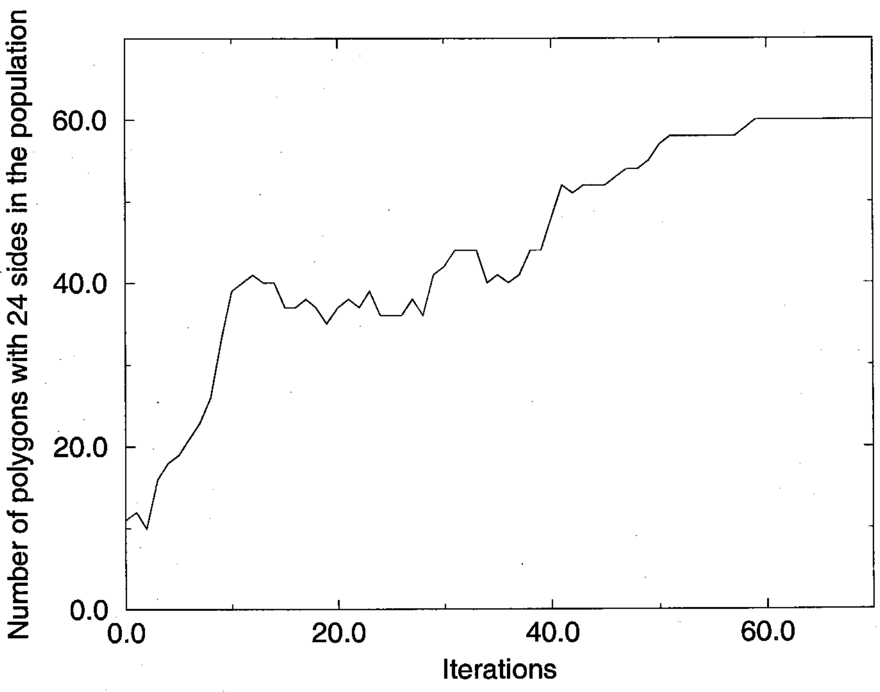


Figure 6.9: Convergence of number of polygons with 24 sides to the population size

12 minutes. The results of synthesis are shown in Figure 6.7 and Table 6.3. In Figure 6.7, the lower-right frame shows the target shape, and the other frames show the best candidate mask-layouts (the dark polygons) at five different iterations during the synthesis loop, and the etched (simulated) 3-D shapes. The convergence of the etched shapes to the target shape can be easily observed. Detailed information about these five iterations is shown in Table 6.3. The shape and size mismatch values between the etched shape and the target shape for each iteration decreased as the synthesis proceeded. At iteration 50, the etched shape of the best mask-layout is very close to the target square mesa, and the corresponding mask-layout can be taken as the optimum mask-layout for the square mesa. The compensation pattern at the convex corners can be observed.

During the synthesis, shape mismatch and size mismatch values were calculated for each candidate mask-layout in each generation during the performance evaluation. The average shape mismatch value and average size mismatch value were obtained for each generation, and Figure 6.8 shows the convergence of the average shape mismatch and average size mismatch along with the iterations. As shown in Table 6.3, during the synthesis, polygons with number of sides between 4 and 28 were explored. The final synthesized mask-layout has 24 sides, and Figure 6.9 shows the convergence of the number of the candidate mask-layouts with 24 sides in each generation to the population size (60) as the synthesis proceeded.

More synthesis runs were conducted to generate alternative mask-layouts for the target shape. Starting with a different initial population of randomly generated polygons, the synthesis can find different mask-layouts as the final solution. Also, the synthesis can be conducted on a subspace of polygons with a specified side number, and in that case the synthesized mask-layout will have a specified number of sides. Various synthesized mask-layouts and the etched shapes are shown in Figure 6.10, and it can be seen that different synthesized compensation structures all can fabricate shapes close to the target shape.

As another test, synthesis was conducted for square mesas with different feature lengths. Previously, the feature length for the square mesa was 500 μm, and the etching depth was 92 μm. To get different aspect ratios for the target shape, the etching depth is unchanged, while different values are used for the square mesa feature length. The synthesized mask-layouts and the etched shapes for different feature lengths are shown in Figure 6.11.

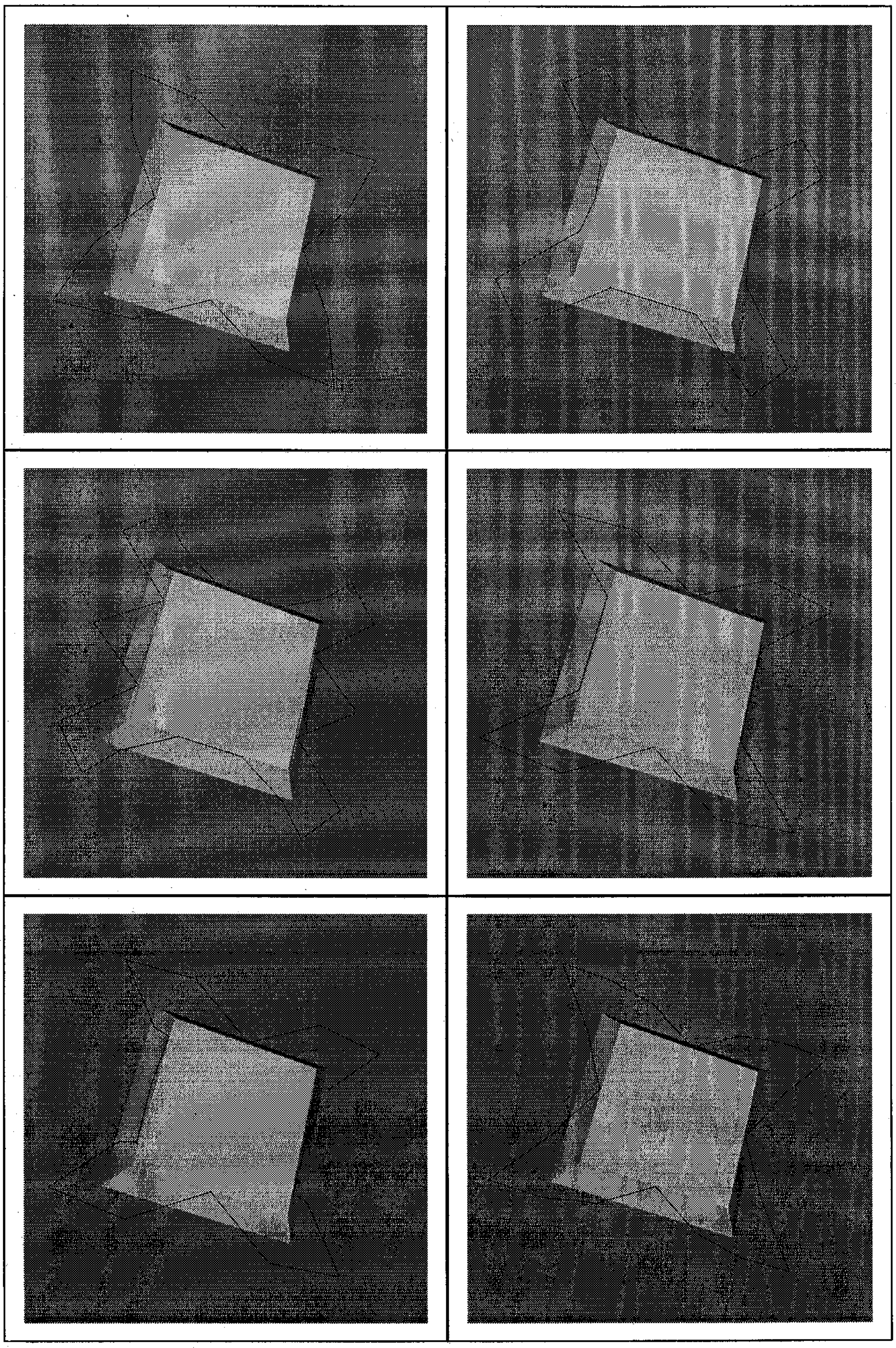


Figure 6.10: Various synthesized mask-layouts for the square mesa

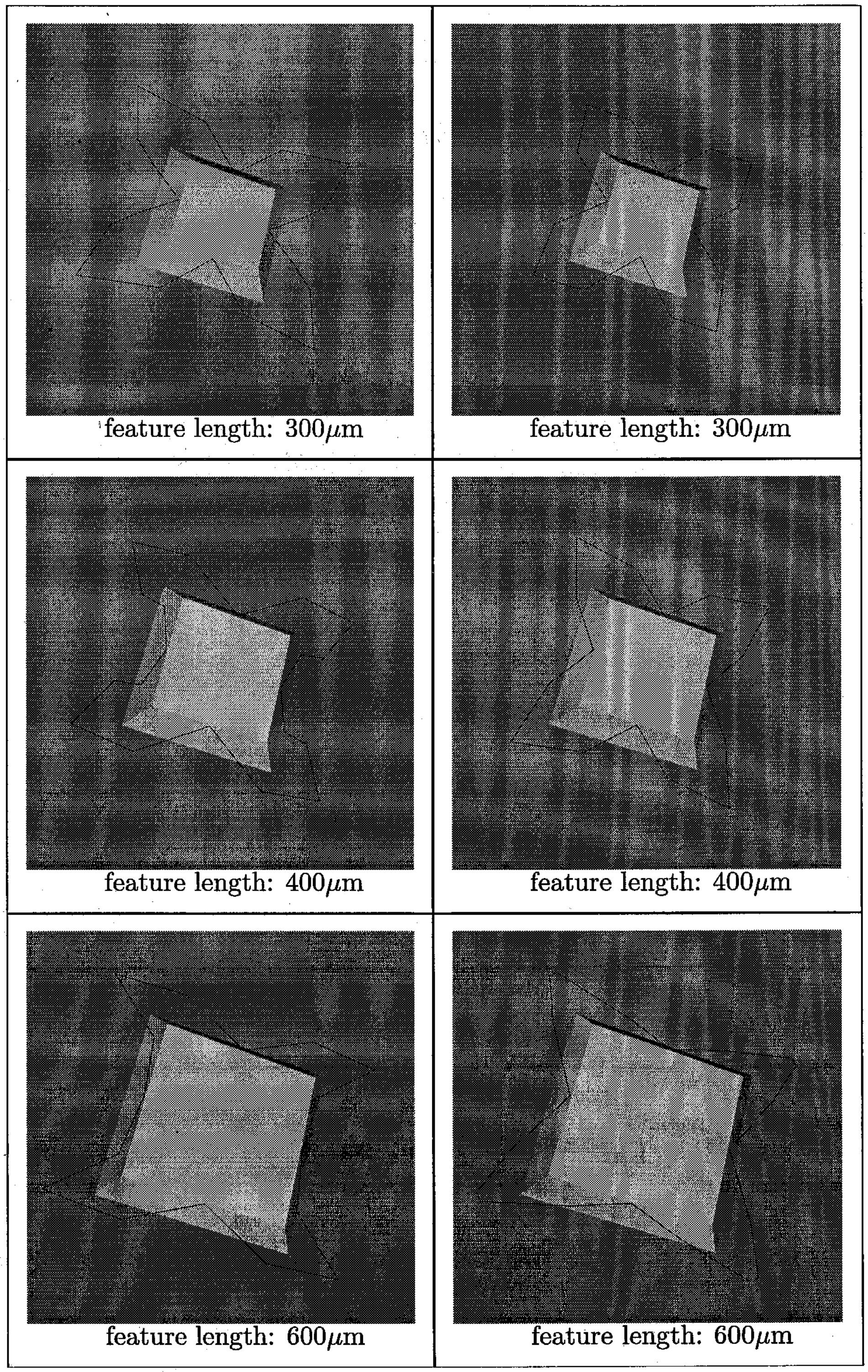

feature length: 300μm
feature length: 300μm
feature length: 400μm
feature length: 400μm
feature length: 600μm
feature length: 600μm

Figure 6.11: Synthesized mask-layouts for the square mesa with different feature lengths

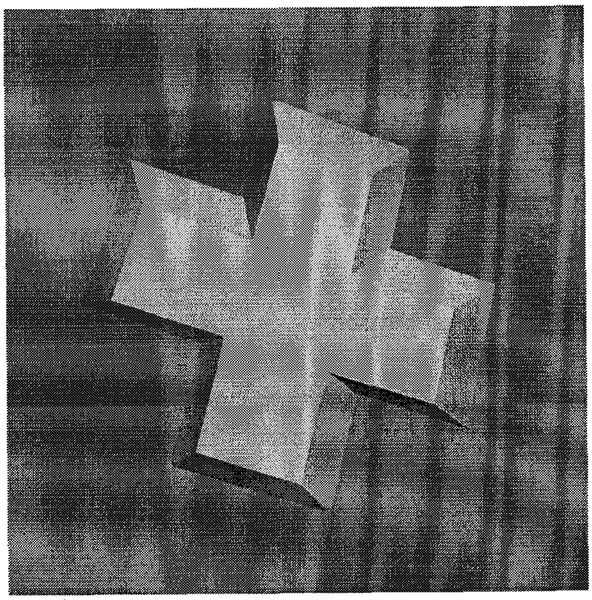


Figure 6.12: A cross groove

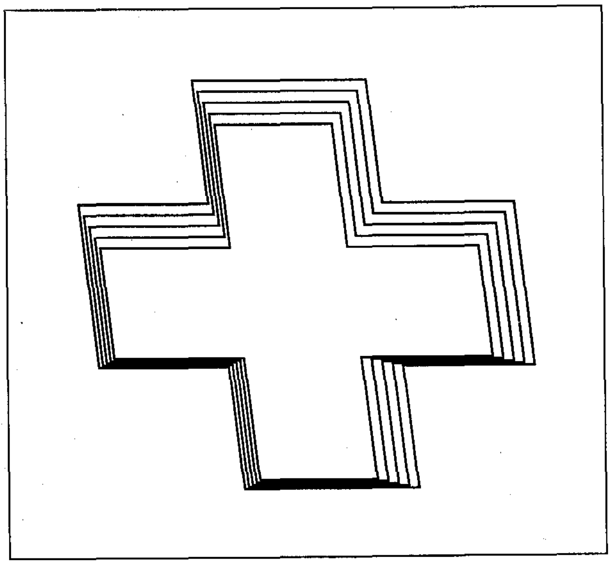


Figure 6.13: A cross groove represented by polygon layers

Parameter	Value
Population Size	70
Iteration Number	80
Minimum Polygon Side Number	4
Maximum Polygon Side Number	48

Table 6.4: Evolution parameters for GA

Table 6.5: Synthesis data showing convergence

Iteration	Side Number	Shape Mismatch	Size Mismatch
1	36	0.313	0.248
7	24	0.284	0.213
15	32	0.143	0.119
22	28	0.128	0.041
55	28	0.024	0.008

6.4 Cross Groove

A second synthesis test was conducted on a cross groove shown in Figure 6.12, and the layer representation of the target shape is shown in Figure 6.13. The KOH etch rate data shown in Table 6.1 were used here also, and the goal of the synthesis was to find proper mask-layout for the cross groove. Similar to the square mesa, the cross groove target shape consists of five polygon layers, and all the vertical intervals are equal to 23 μm. All the edges of the polygons are aligned with <110> direction, and all the sidewalls are <111> planes. The length of the groove is 1000 μm, and the width is 400 μm (all measured on the top layer).

The synthesis for the cross groove is more challenging than the square mesa although in both cases the main effort is focused on forming the compensation pattern for the convex corners. Since the total area available for the compensation structure is limited by the cross boundary, the synthesized compensation pattern needs to be small enough to satisfy the area constraint. This becomes more challenging for cross grooves with higher aspect ratios.

After some preliminary tuning, the evolution parameters used in the Genetic Algorithm were set as shown in Table 6.4.

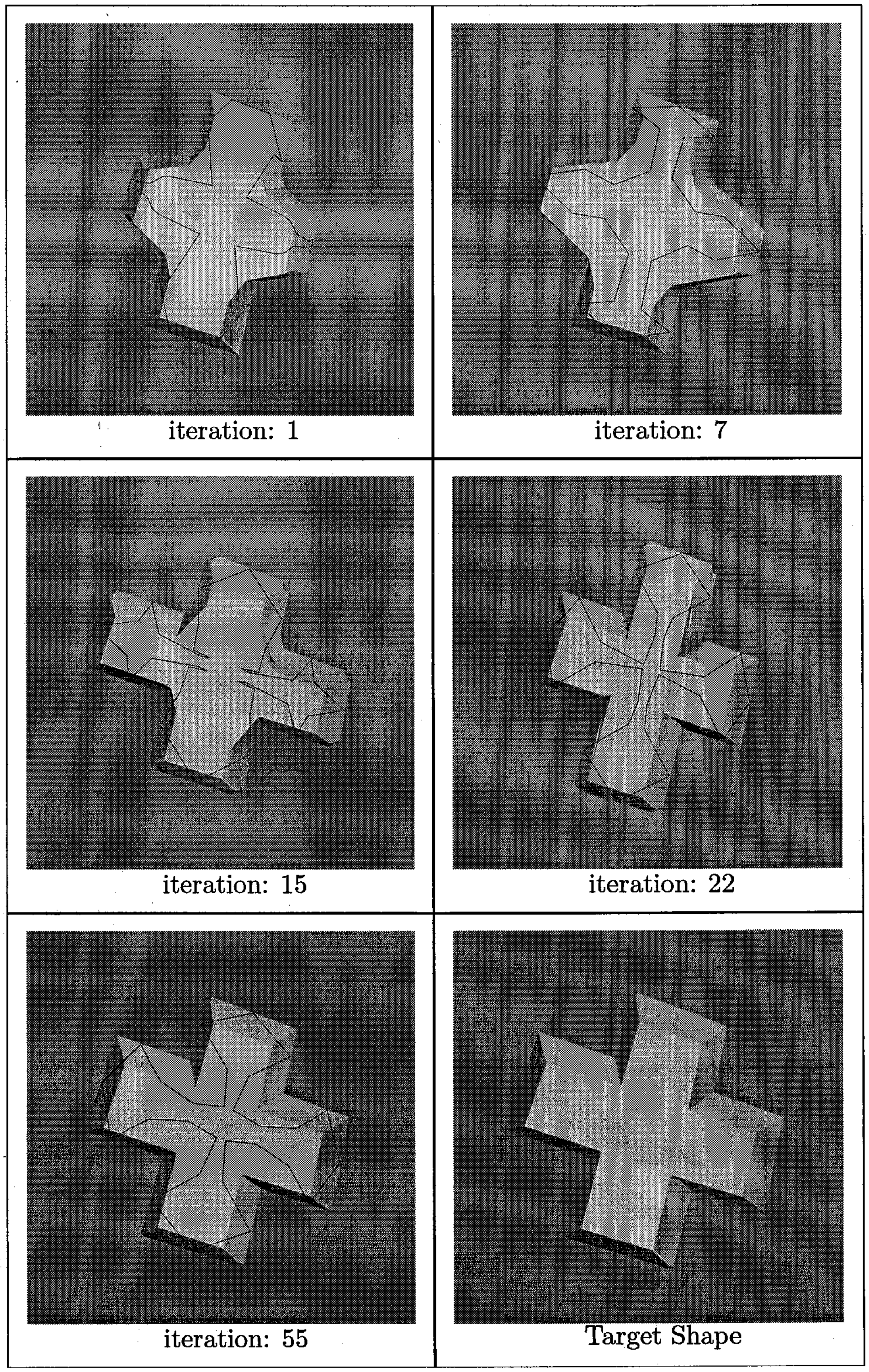


Figure 6.14: Synthesis results: mask-layouts and simulated shapes

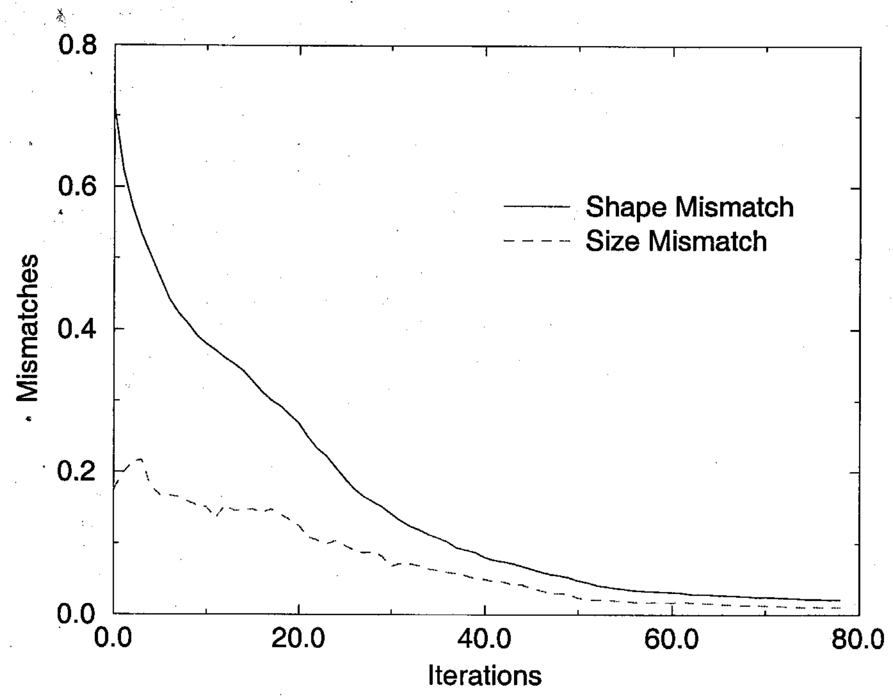


Figure 6.15: Convergence curves of shape and size mismatch values

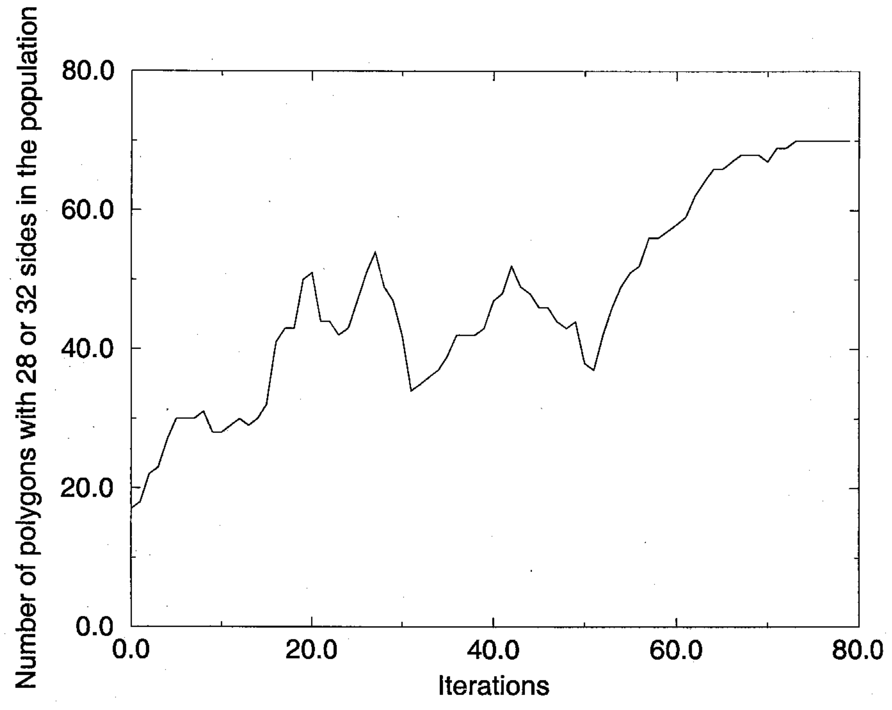


Figure 6.16: Convergence of number of polygons with either 24 or 28 sides to the population size

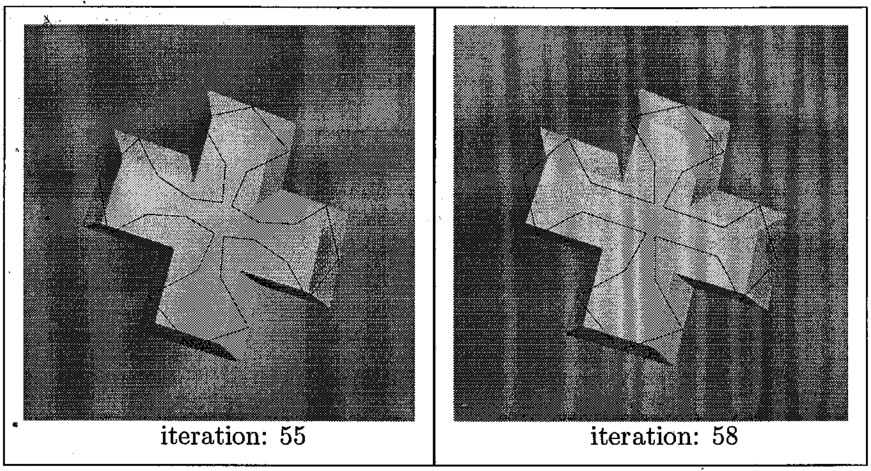


Figure 6.17: The best mask-layout at iteration 55 and iteration 58

The synthesis task run took about 15 minutes on a Sun Ultra 10 workstation. The results of synthesis are shown in Figure 6.14 and Table 6.5. In Figure 6.14, the lower-right frame shows the target shape, and the other frames show the best candidate mask-layouts (the dark polygons) at five different iterations during the synthesis loop, and the etched (simulated) 3-D shapes. The convergence of the etched shapes to the target shape can be easily observed. Detailed information about these five iterations is shown in Table 6.5. The shape and size mismatch values between the etched shape and the target shape for each iteration decreased as the synthesis proceeded. At iteration 55, the etched shape of the best mask-layout is very close to the target square mesa, and the corresponding mask-layout can be taken as the optimum mask-layout for the square mesa. The compensation pattern at the convex corners can be observed.

The average shape mismatch value and average size mismatch value were calculated for each generation, and Figure 6.15 shows the convergence of the average shape mismatch and average size mismatch along with the iterations. As shown in Table 6.5, during the synthesis, polygons with number of sides between 4 and 48 were explored. Examining the mask-layouts in all generations, it was found that the searching (evolution) did not converge to a subspace of polygons with a particular side number, as the case with the synthesis for the square mesa, but converged to two subspaces of polygons. To be more specific, at the late stage of the evolution, most of the polygons have side number of either 24 or 28, and the search effort was focused on the subspace of polygons with 24 sides and subspace of polygons with 28 sides. Figure 6.16 shows the convergence of the number of the candidate mask-layouts with either 24 sides or 28 sides in each generation to the population size (70) as the synthesis proceeded. The Genetic Algorithm explored promising mask-layouts with different topologies to search for the optimum solution for the target shape, and this actually is one of the characteristics of GA: multiple optima can be searched simultaneously by GA. For example, Figure 6.17 shows the best mask-layout at iteration 55, which has 28 sides, and the best mask-layout at iteration 58, which has 24 sides, and both mask-layouts generate shapes fairly close to the target shape.

More synthesis runs were conducted to generate alternative mask-layouts for the target shape, and various synthesized mask-layouts and the etched shapes are shown in Figure 6.18.

As another test, syntheses were conducted for cross grooves with different feature lengths. Previously, the feature length for the cross groove was 1000 μm (length) and 400 μm (width), and the etching depth was 92 μm. To get different aspect ratios for the target shape, the etching depth is unchanged, while different values are used for the cross groove feature length (groove length and width). The synthesized mask-layouts and the etched shapes for different feature lengths are shown in Figure 6.19.

6.5 Mask-layout and Process Synthesis

In the previous two examples, a single wet etching step was used in the fabrication, and only the mask-layout needed to be synthesized for a given target shape. In many cases, the fabrication process flow itself needs to be synthesized. As described in Chapter 3, complex three-dimensional shapes can be generated using fabrication procedures such as multi-step wet etching process [43, 63]. For a multi-step wet etching process, the etchant sequence and the etch duration for each etchant need to be determined by the designer, and for a given target shape, it is more challenging to design a right formula (process flow) compared to single-step etching because of the large number of different possibilities. A mask-layout and process synthesis methodology is badly needed to help the designer design the mask-layout and the fabrication process flow for a desired device shape.

To demonstrate the capability of the GA approach to the mask-layout and process synthesis, a synthesis test was conducted to synthesize mask-layout and process flow for a two-step wet etching process. There were three different etchants available (the etch rate data are shown in Table 6.6), while the fabrication procedure that was considered was two-step wet etching. That means two etchants can be applied in sequence to generate the final device shape. For a given target shape, the goal of the synthesis was to find a proper mask-layout and also the right process flow (*i.e.*,

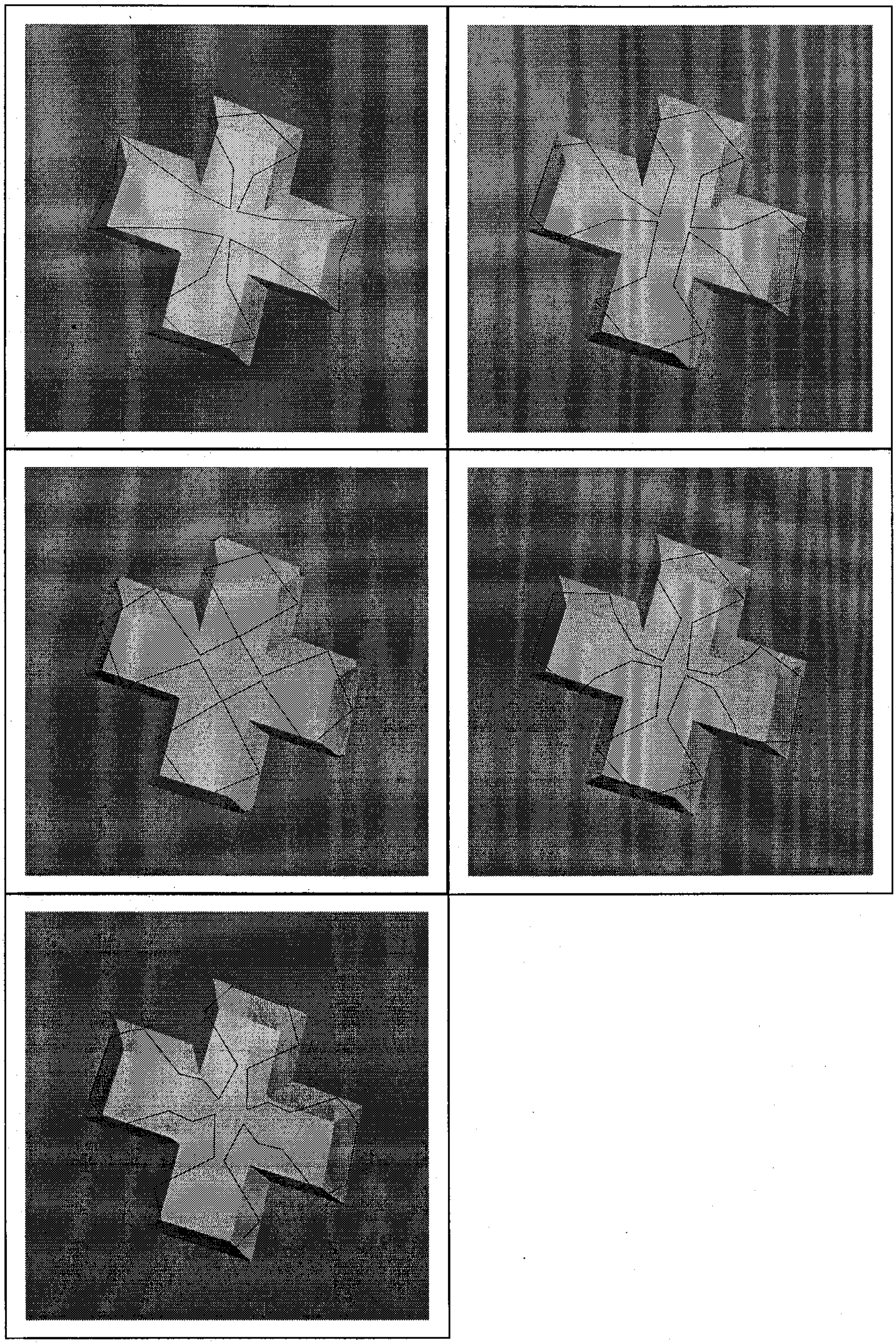


Figure 6.18: Various synthesized mask-layouts for the cross groove

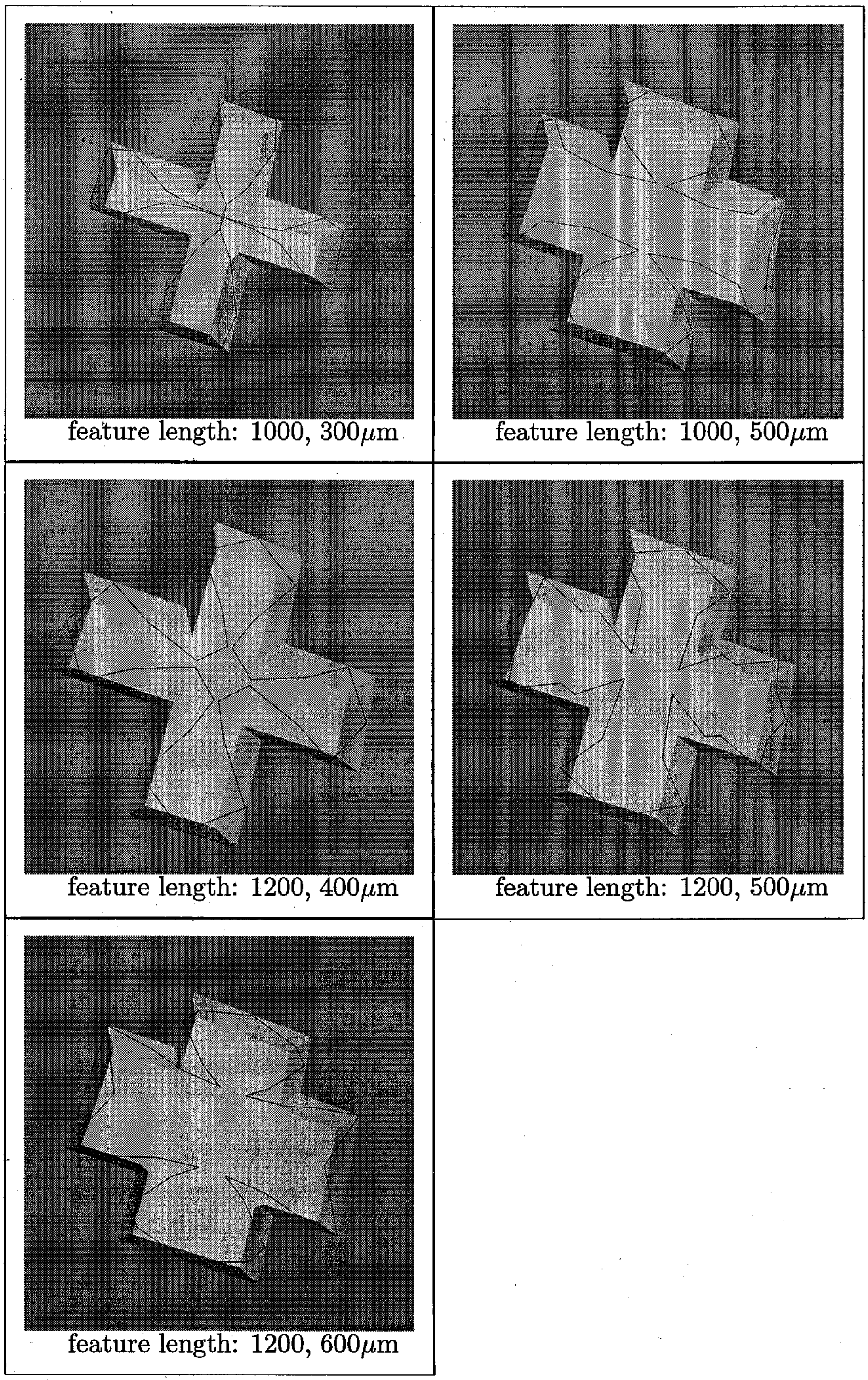

feature length: 1000, 300μm
feature length: 1000, 500μm
feature length: 1200, 400μm
feature length: 1200, 500μm
feature length: 1200, 600μm

Figure 6.19: Synthesized mask-layouts for the cross groove with different feature lengths

Orientation	Etchant No. 1 (μm/hr)	Etchant No. 2 (μm/hr)	Etchant No. 3 (μm/hr)
100	5	5	5
110	5	5	1
311	1	6	unknown
111	6	1	unknown

Table 6.6: Etching rates for three different etchants

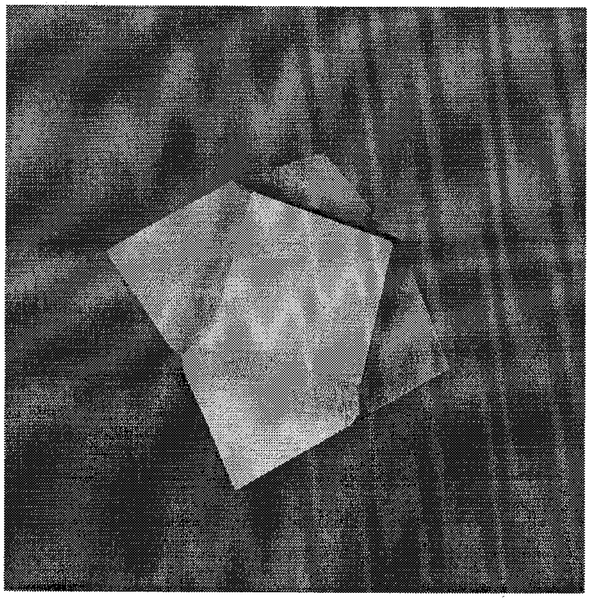


Figure 6.20: The target shape

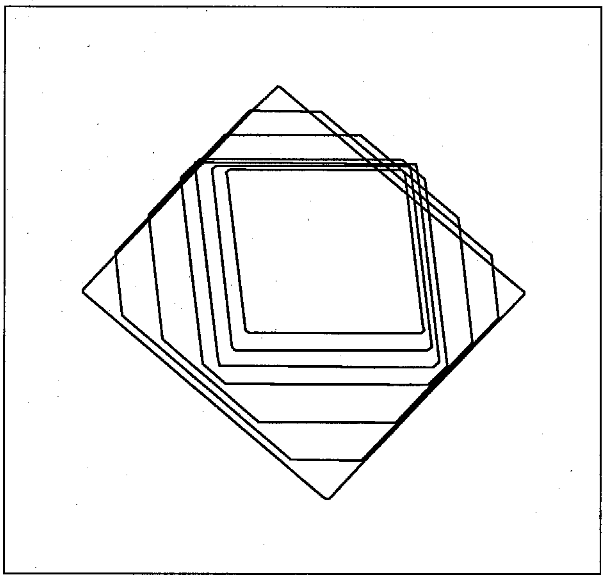


Figure 6.21: The target shape represented by polygon layers

Parameter	Value
Population Size	80
Iteration Number	60
Minimum Polygon Side Number	4
Maximum Polygon Side Number	32

Table 6.7: Evolution parameters for GA

which two etchants to use, the order they will be applied, and the etch time for etch etchant). For the process flow synthesis, the coding scheme and genetic operations described in section 5.6 were used to synthesize the etchant sequence and etch durations. The target shape is shown in Figure 6.20, and the layer representation of the target shape is shown in Figure 6.21.

Using the evolution parameters shown in Table 6.7, the synthesis task run took about 35 minutes on a Sun Ultra 10 workstation. The results of synthesis are shown in Figure 6.22 and Table 6.8. In Figure 6.22, the lower-right frame shows the target shape, and the other frames show the best candidate mask-layouts (the dark polygons) at five different iterations during the synthesis loop, and the etched (simulated) 3-D shapes. The convergence of the etched shapes to the target shape can be easily observed. Detailed information about these five iterations is shown in Table 6.8. The shape and size mismatch values between the etched shape and the target shape for each iteration decreased as the synthesis proceeded. The etchant sequence converged to (#2, #1), and the etch

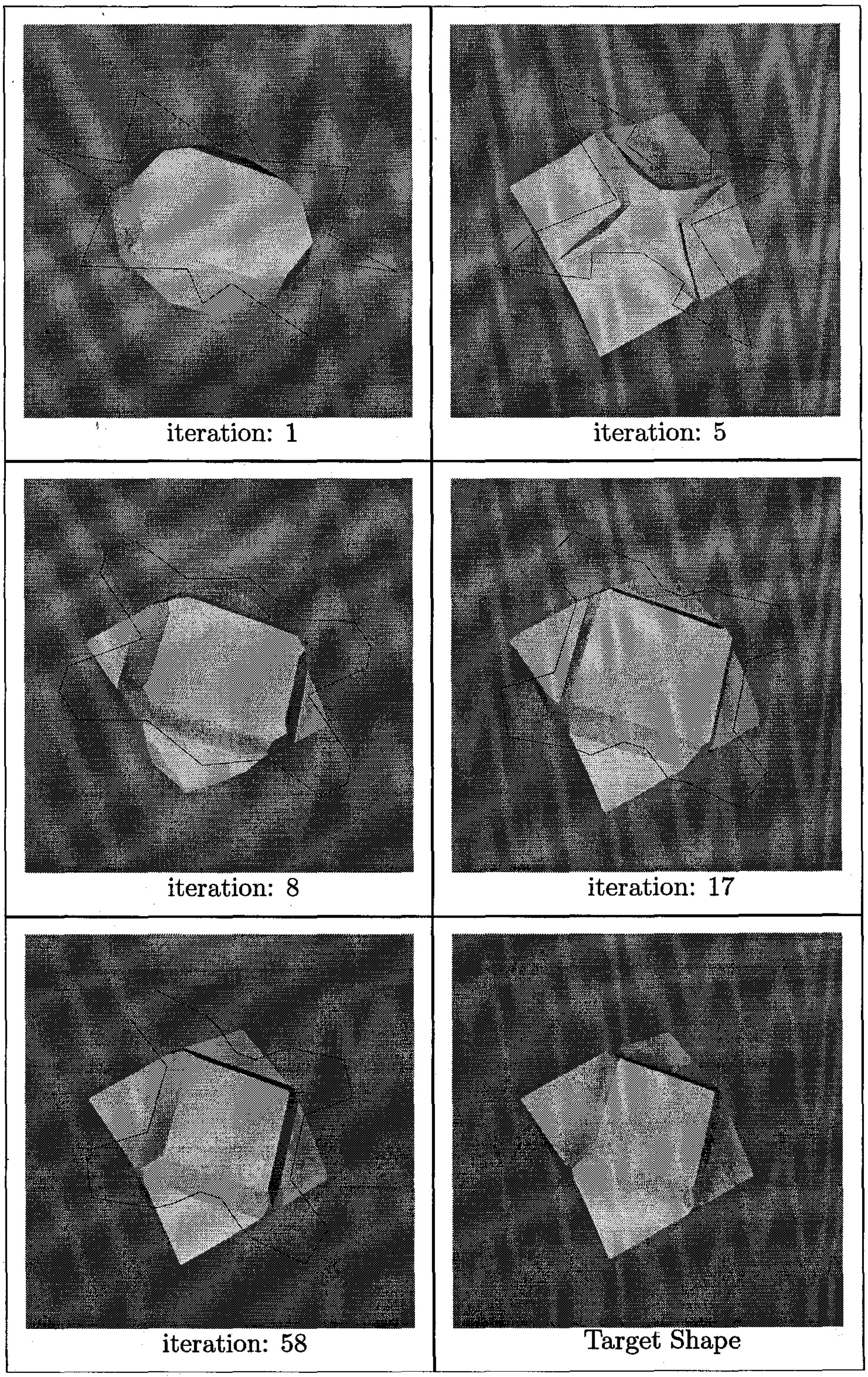


Figure 6.22: Synthesis results: mask-layouts and simulated shapes

Table 6.8: Synthesis data showing convergence

Iteration	Etchant Sequence	Etch Times	Side Number	Shape Mismatch	Size Mismatch
1	3, 2	0.18, 5.82	16	0.339	0.130
5	1, 1	5.91, 0.09	20	0.165	0.088
8	2, 1	3.90, 2.10	24	0.117	0.061
17	2, 1	3.02, 2.98	24	0.089	0.075
58	2, 1	2.86, 3.14	24	0.018	0.018

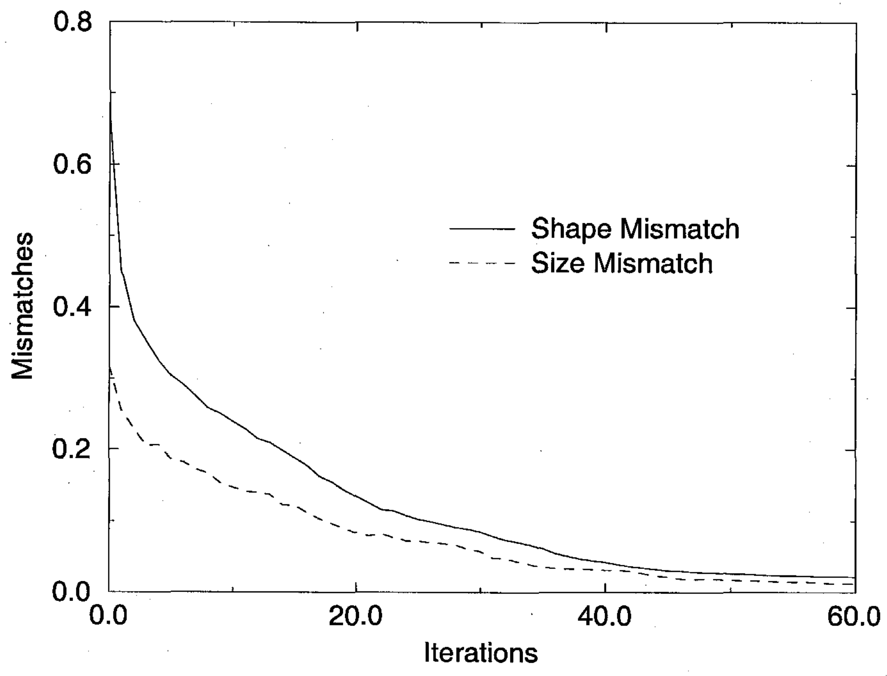


Figure 6.23: Convergence curves of shape and size mismatch values

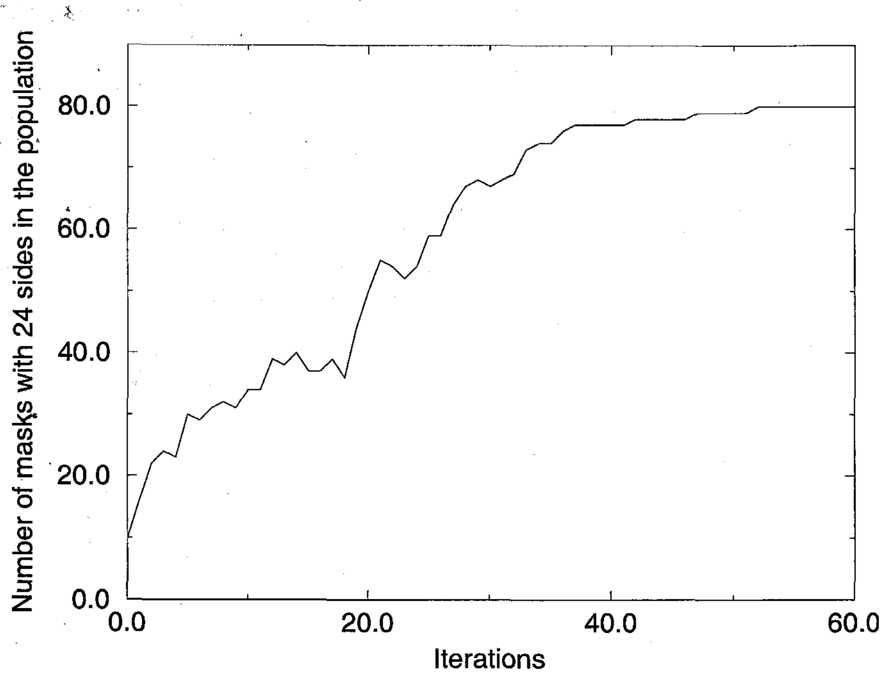


Figure 6.24: Convergence of number of polygons with 24 sides to the population size

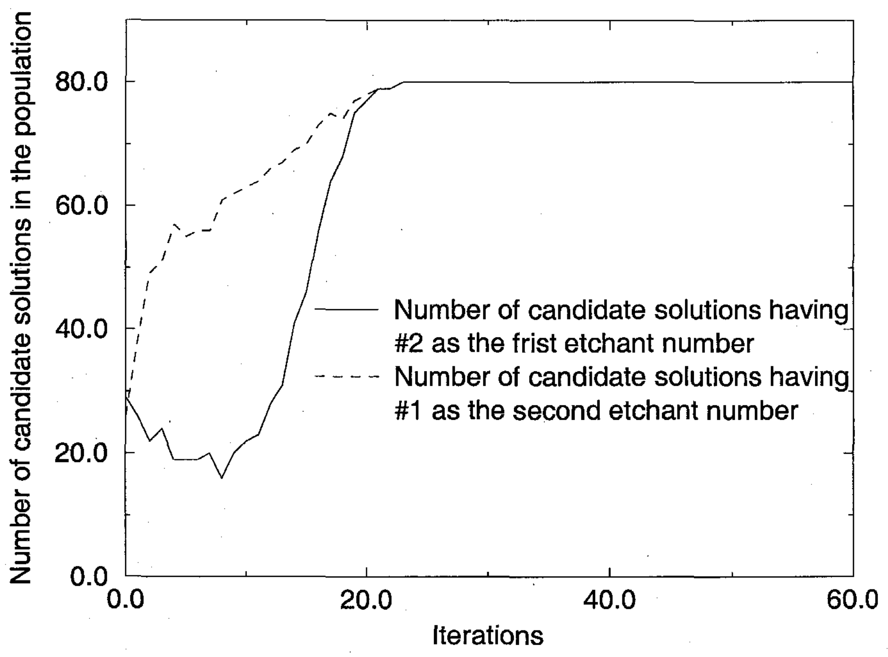


Figure 6.25: Convergence of number of candidate solutions having #2 and #1 as the first and second etchant numbers to the population size

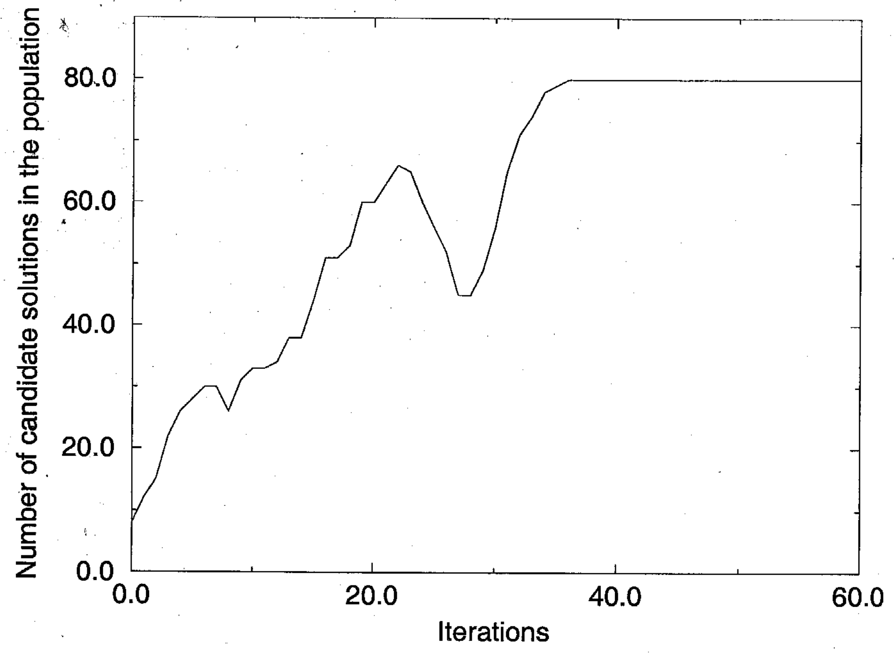


Figure 6.26: Convergence of number of candidate solutions having the first etch time between 2.5 hours and 3.5 hours to the population size

times converged to about (3 hours, 3 hours). At iteration 58, the etched shape of the best mask-layout and process flow is very close to the target shape. The final solution can be stated as follows: using the best mask-layout at iteration 58, apply etchant #2 for 2.86 hours, and then etchant #1 for 3.14 hours, a device shape fairly close to the target shape can be fabricated.

The average shape mismatch value and average size mismatch value were calculated for each generation, and Figure 6.23 shows the convergence of the average shape mismatch and average size mismatch along with the iterations. As shown in Table 6.8, during the synthesis, polygons with number of sides between 4 and 32 were explored. The final synthesized mask-layout has 24 sides, and Figure 6.24 shows the convergence of the number of the candidate mask-layouts with 24 sides in each generation to the population size (80) as the synthesis proceeded. Similarly, the number of candidate solutions having #2 as the first etchant number and #1 as the second etchant number was counted for each iteration, and the convergence of the first and second etchant numbers to #2 and #1 is shown in Figure 6.25. For etch times, the synthesis data showed that the first etch time converged to around 3.0 hours. The number of candidate solutions having the first etch time between 2.5 hours and 3.5 hours was counted for each iteration, and the convergence of this number to the population size is shown in Figure 6.26.

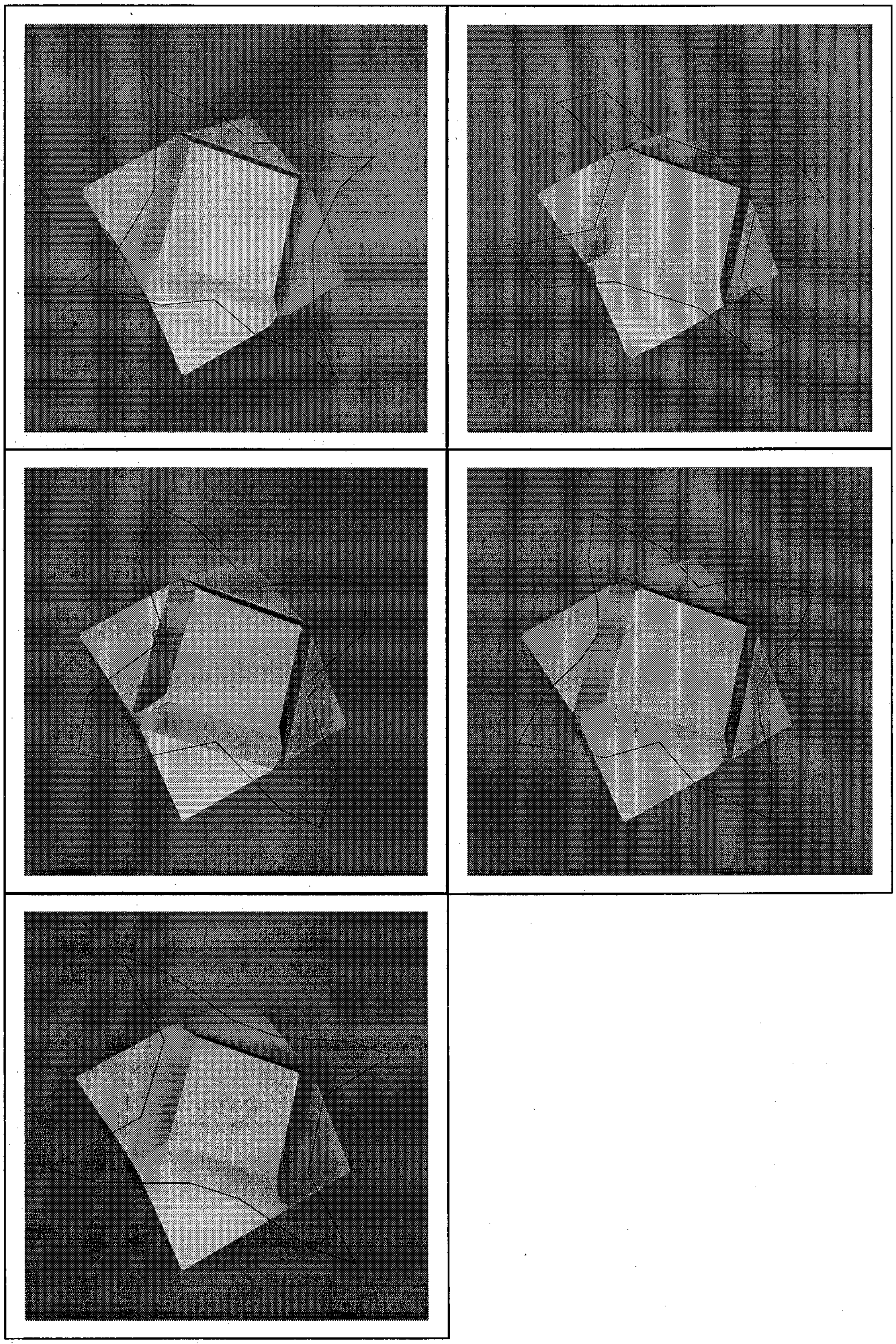


Figure 6.27: Various synthesized mask-layouts for the target shape

More synthesis runs were conducted to generate alternative mask-layouts for the target shape, and various synthesized mask-layouts and the etched shapes are shown in Figure 6.27. Figure 6.27 shows that different compensation structures all can fabricate shapes close to the target shape. Please note that although different synthesis runs generated different optimum mask-layouts, all synthesis runs generated the same etchant sequence (#2 and #1) and close etch durations (around 3 hours, 3 hours).

6.6 Summary

In this chapter, the Genetic Algorithm mask-layout and process synthesis approach was tested for bulk wet etching. The first and second tests were to synthesize compensation structures for a single-step wet etching given a square mesa and a cross groove as target shape, and different compensation patterns have been synthesized. The third test was about synthesis of both the mask-layout and the process flow for a two-step wet etching process. For a given target shape, the algorithm was able to synthesize not only a proper mask-layout, but also a process flow (the etchant sequence and the etch time for each etchant) to fabricate the target shape. Synthesis results and data were shown to illustrate the convergence characteristics of the Genetic Algorithm, and the feasibility of the GA mask-layout and process synthesis approach was verified.

Chapter 7

Robust Design

7.1 Overview

In the previous chapters, a Genetic Algorithm approach for MEMS mask-layout and process synthesis has been introduced, and proper mask-layout and process flow can be automatically generated for given target shape. However, during the search for optimal solutions (mask-layouts and process flows), no manufacturing variations (*e.g.*, mask-layout misalignment, temperature variations) or modeling inaccuracy (*e.g.*, etch rate modeling inaccuracy) have been considered in the evaluation of the candidate solutions. Thus, although the final solutions synthesized by the Genetic Algorithm have optimal performance in the simulation, the optimal performance is based on an assumed perfect manufacturing environment and modeling accuracy which are practically impossible in actual fabrication. The inevitable manufacturing variations and modeling inaccuracy may seriously reduce the performance of the solutions generated by the Genetic Algorithm synthesis method. For example, for a square mesa, the Genetic Algorithm mask-layout synthesis method generates a mask-layout shown as a dark polygon in Figure 7.1 together with the fabricated 3-D shape. The fabricated shape matches the target square mesa very well. However, if during the fabrication, the mask-layout is misaligned with the crystalline orientation of the silicon wafer by 3°, a 3-D shape shown in Figure 7.2 will be generated. The mismatch between this shape and the target mesa is significant, and this device shape may not be acceptable.

Since noise (variations) always exists in the fabrication procedure, solutions robust to the variations are highly desired, and robust design is the focus of this chapter.

For robust design of mask-layout and process flow, the Genetic Algorithm approach is extended by incorporating noise (variations) into the Genetic Algorithm search iterations. Figure 7.3 schemat-

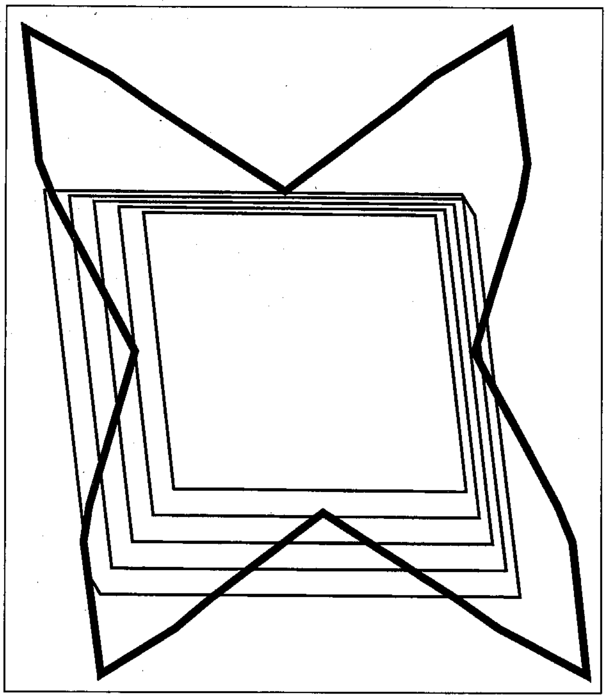


Figure 7.1: Synthesized mask-layout for a square mesa

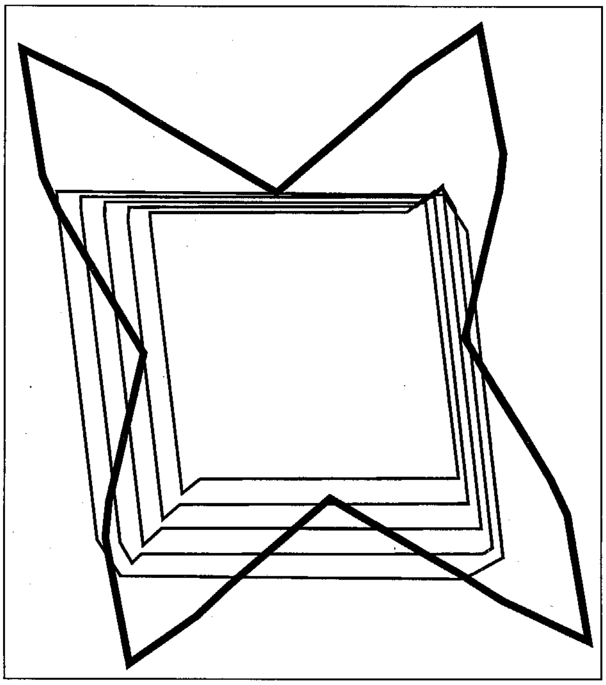


Figure 7.2: Fabricated shape when the mask has a 3° misalignment

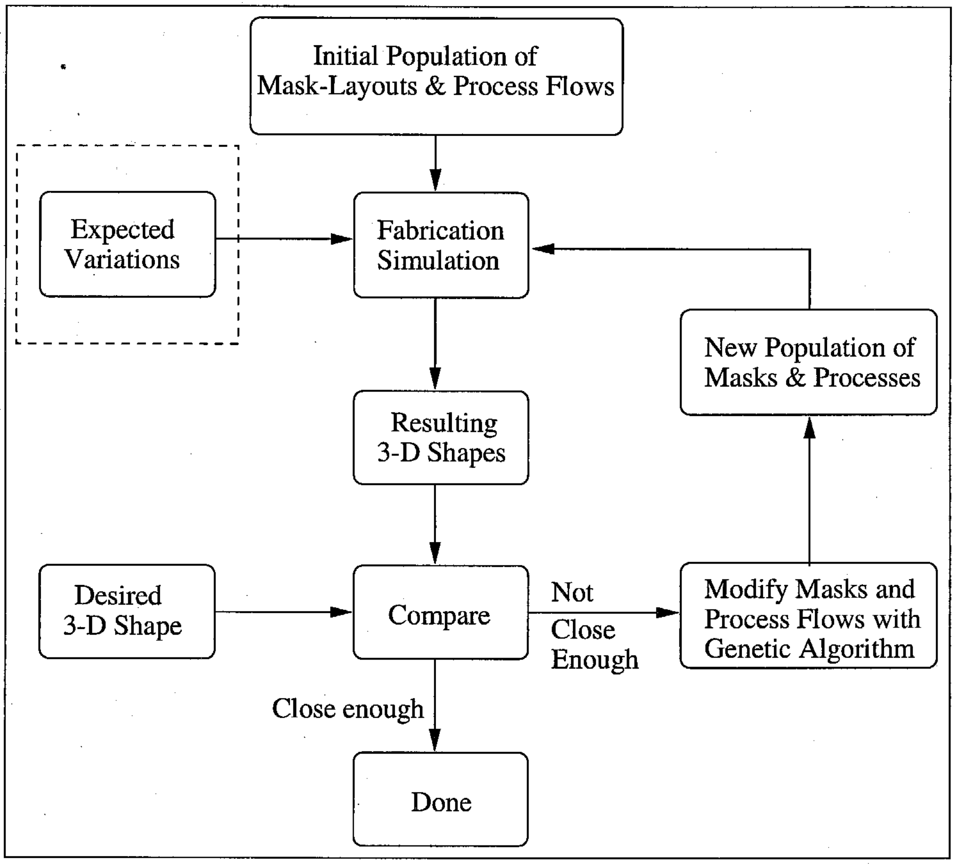

Initial Population of Mask-Layouts & Process Flows
Expected Variations
Fabrication Simulation
New Population of Masks & Processes
Resulting 3-D Shapes
Desired 3-D Shape
Compare
Not Close Enough
Modify Masks and Process Flows with Genetic Algorithm
Close enough
Done

Figure 7.3: A schematic representation of a Genetic Algorithm MEMS synthesis technique for robust design

ically demonstrates how noise factors are integrated into the design process using the Genetic Algorithm approach to design a solution which is robust to manufacturing variations and modeling inaccuracy. The difference between the robust design scheme and the original Genetic Algorithm synthesis scheme can be observed by comparing Figure 7.3 with Figure 5.1.

In the following sections, first an introduction of the background in robust design area and the Taguchi Method is given. Then the concept of Signal-to-Noise (S/N) ratio in the Taguchi Method is used to modify the Genetic Algorithm synthesis approach for robust design. The implementation details and a robust design example for mask misalignment are described. Finally, an alternative approach to robust design using a specific Genetic Algorithm sampling and evaluation scheme is introduced which can save computational cost when there are multiple noise sources. A robust mask-layout design example for etch rate variations is described to illustrate the algorithm.

7.2 Background

The goal of mask-layout and process flow synthesis is to design a fabrication process (mask-layout can be viewed as part of the process) for a given geometry or performance of a MEMS device. There are always difficult-to-control variations during the manufacturing and operational life of a process or device. Neglecting these variations during the product/process design sometimes results in a product or process which when put in operation exhibits unexpected or unacceptable performance. Robust design is an important method for improving product or process design by making the output response insensitive (robust) to uncontrolled variations (noise). The primary goal of robust design is to develop stable products and processes that exhibit minimum sensitivity to uncontrollable operational fluctuations.

7.2.1 Product/Process Design

A block diagram representation of a product (or process) is shown in Figure 7.4. The response of the product is denoted by y. The response could be the output of the product or some other suitable characteristic. The response considered for the purpose of optimization in a robust design experiment is called a quality characteristic, or performance parameter.

A number of parameters can influence the quality characteristic or response of the product. These parameters can be classified into the following three classes:

1. Signal factors (M). These are the parameters set by the user of the product to express the

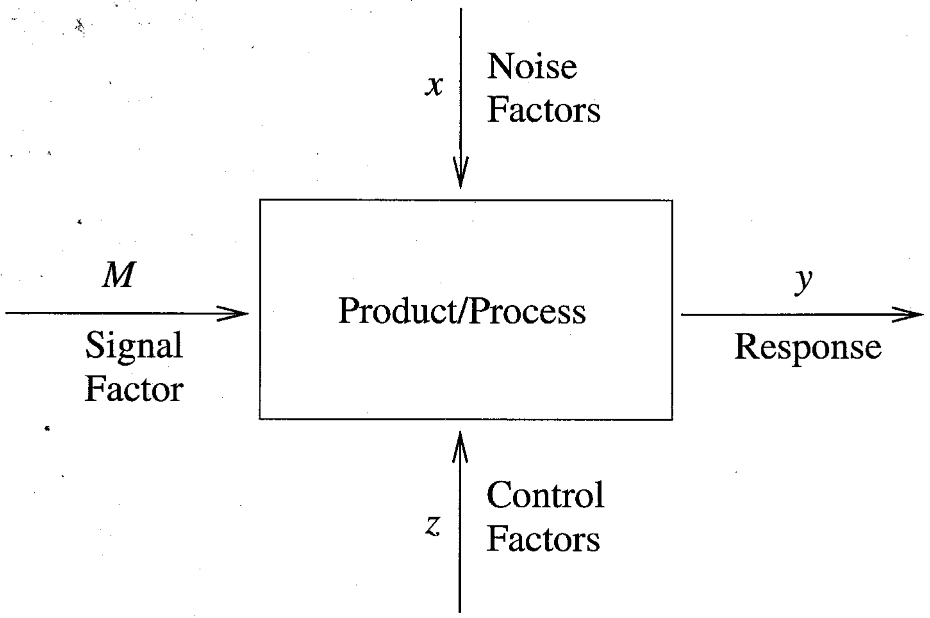


Figure 7.4: Block diagram of a product/process: P Diagram

intended value for the response of the product.

2. Noise factors (x). Certain parameters cannot be controlled by the designer and are called noise factors. Noise factors cause the response y to deviate from the target specified by the signal factor M and lead to quality loss.

3. Control factors (z). These are parameters that can be specified freely by the designer. Control factors are also called design parameters, and it is the designer's responsibility to determine the best values of these parameters.

Identifying important responses, signal factors, noise factors and control factors in a specific project is an important task. For example, for the mask-layout and process synthesis problem, the goal is to design a mask-layout and process flow for a desired target device shape. The target shape is the signal factor; noise factors include mask misalignment, temperature variation, *etc.*; control factors are the mask-layout and the process flow; the fabricated device shape is the response, and the match result between the fabricated shape and the target shape can be taken as the performance parameter.

The three major steps in designing a product or a process are concept design, parameter design, and tolerance design. Concept design deals with selection of product architecture or process technology. In this step, the designer examines a variety of architectures and technologies for achieving the desired function of the product and selects the most suitable ones for the product. Tolerance design is about selection of the optimum values of the tolerance factors to balance the improvement in quality loss against the increase in the unit manufacturing cost. Tolerance design should be performed only after sensitivity to noise has been minimized through parameter design. Parameter design is the design phase between concept design and tolerance design. The optimum values of the control factors are determined in parameter design to maximize robustness. In parameter design, designer determines the best settings for the control factors (*i.e.*, the settings that minimize quality loss) that do not affect manufacturing cost. Thus, designer must minimize the sensitivity of the function of the product or process to all noise factors and also get the mean function on target.

Parameter design improves quality of product or process without increasing the manufacturing cost, and is the most inexpensive way to improve quality. Robust Design and its associated methodology focus on parameter design.

7.2.2 Robust Design

Variations in noise factors (uncontrollable parameters) always exist. It is often difficult, if not impossible, to eliminate sources of variation that contribute to a product's poor performance. It is, however, more practical to develop a product or process least sensitive to these variations. The fundamental principle in robust design is to improve the quality of a product by minimizing the effects of variation without eliminating these causes.

Robustness refers to the ability of products or processes to perform consistently under varying operational conditions. During robust design, a designer seeks to determine the control parameter settings that produce desirable values of the performance mean, while at the same time minimizing the variance of the performance. Robust design, then, is a multiobjective and nondeterministic approach, and is concerned with both the performance mean and the variability that result from uncertainty (represented through noise variables). In this setting, sensitivity analysis is concerned with both the mean and variance of the performance. The performance variation is often minimized at the cost of sacrificing the best performance, and therefore the tradeoff between the aforementioned two aspects cannot be avoided.

Many different robust design approach have been developed, including the Taguchi method and extensions [88, 92, 112] (which are outlined in next section), Response Surface Methodology and Compromise Programming approach [20, 21] (which addresses the multiple aspects of the objective in robust design), Simulated Variance Optimization [89], and Genetic Algorithms for robust design [36, 40, 81, 116].

Most optimization methods for robust design involve optimization of a statistical estimate of a performance parameter obtained by experiments or computer simulation. In this approach, the noise factors are treated as random variables with assigned probability distributions. The design in question is defined by one or more equations giving a performance parameter as a function of all or some of the design parameters (*i.e.*, a given design point) and noise factors, interpreted here as the expected values of the random variables. For a given design point (a set of design parameters), a large sample of values of the performance parameter may be obtained by repeated sampling of the noise factors from their assigned distributions by use of random number generators. The resulting data can then be used to obtain the expected value and variance of the performance parameter and therefore the value for the statistical estimate at the given design point. The optimum design parameters are then obtained at those corresponding to a minimum of the computed estimate, determined by means of some non-linear optimization algorithm in the presence of constraints.

7.3 Taguchi Method and S/N Ratio

The Taguchi robust design method has been widely used to design quality into products and processes. Using this method, the quality of a product is improved by minimizing the effect of the causes of variation without eliminating the causes. Reference [92] gives a complete description of the method, and here the Signal-to-Noise ratio (S/N ratio), the key concept in the Taguchi method, is briefly outlined.

Quality Loss Function

The deviation of performance from the ideal is defined as Quality Loss in the Taguchi method. The quadratic loss function can meaningfully approximate the quality loss in most situations. Let y be the quality characteristic of a product and m be the target value for y. According to the quadratic loss function, the quality loss is given by

$$L(y) = k(y - m)^2$$

where k is a constant called *quality loss coefficient*.

Because of the noise factors, the quality characteristic y of a product or process varies. Let

$y_1, y_2, \ldots, y_n$ be n representative measurements of the quality characteristic y, then the average quality loss, Q, resulting from this product is given by

$$\begin{aligned} Q &= \frac{1}{n}[L(y_1) + L(y_2) + \ldots + L(y_n)] \\ &= \frac{k}{n}[(y_1 - m)^2 + (y_2 - m)^2 + \ldots + (y_n - m)^2] \\ &= k[(\mu - m)^2 + \frac{n-1}{n}\sigma^2] \end{aligned}$$

where μ and σ^2 are the mean and the variance of y, respectively, as computed by

$$\mu = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{7.1}$$

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(y_i - \mu)^2 \tag{7.2}$$

When n is large,

$$Q = k[(\mu - m)^2 + \sigma^2] \tag{7.3}$$

Thus, the average quality loss has the following two components:

1. $k(\mu - m)^2$ resulting from the deviation of the average value of y from the target;

2. $k\sigma^2$ resulting from the mean square deviation of y around its own mean.

Equation (7.3) shows that minimizing the expected quality loss can be achieved by minimizing both the variance σ^2 and the difference between the mean and the target. As illustrated by the Figure 7.5, robust design is always concerned with aligning the peak of the bell shaped response distribution with the targeted quality (optimizing the mean performance), and making the bell shaped curve thinner (minimizing the variance σ).

S/N Ratio

One of the key features of the Taguchi method is the use of Signal-to-Noise ratio (S/N ratio) to transform the performance characteristic in the optimization process.

In the Taguchi method, the design metric used as the performance measure (performance statistic) usually takes the form of a logarithmic function based on the mean square deviation of product

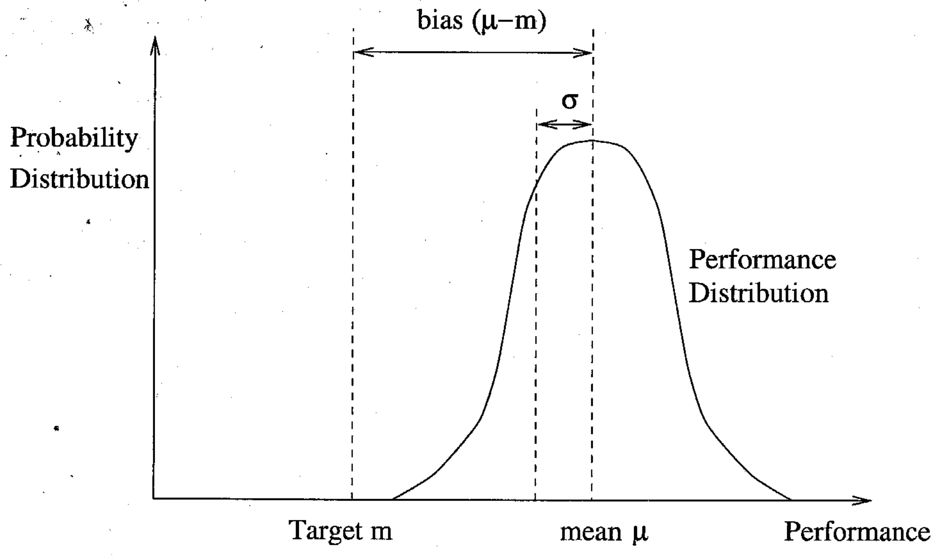


Figure 7.5: Quality distribution in robust design

responses, and is called Signal-to-Noise ratio (S/N ratio). For a set of design parameters (a design point) $\vec{x}$, the S/N ratio for $\vec{x}$ is defined as

$$S/N \equiv -10\, log[(\sum_{i=1}^{n} y_i - m)^2)/n]$$

where n is the total number of experiments, y_i is the performance parameter for an experiment, and m is the desired target value for performance parameter. $y_i = PP(\vec{x}, \vec{p_i})$, where PP is the performance parameter being considered, and $\vec{p_i}$ is noise parameter for an experiment.

The formula of the S/N ratio is closely related to the quality loss function. Similar to the deduction for average quality loss, it is easy to get

$$S/N \equiv -10\, log[(\mu - m)^2 + \sigma^2] \tag{7.4}$$

For minimization problems (the smaller the better), the following S/N ratio can be constructed by a finite number of experiments:

$$\begin{aligned} S/N &= -10\, log[(\sum_{i=1}^{n} y_i^2)/n] &(7.5)\\ &= -10\, log[\mu^2 + \sigma^2] &(7.6)\end{aligned}$$

Obviously the S/N increase signifies a decrease in the average results (small μ) or improved consistency from one unit to another (small σ), and a combined improvement in the mean result and a reduction in the variability will result in the greatest S/N increase.

In the next section, the Taguchi method with its S/N ratio is applied to the mask-layout synthesis problem with mask misalignment noise, and the Genetic Algorithm synthesis approach will be modified to synthesize mask-layouts robust to the misalignment noise.

7.4 Robust Design with S/N Ratio for Mask-Layout Misalignment

In micromachines, alignment poses more complexity than in IC manufacturing. Not only does one deal with high aspect ratio 3-D features causing problems for alignment systems with low DOF, one also frequently needs to align 3-D features on both sides of the wafer. For commercial silicon wafers, the alignment accuracy (between the flat and the crystal orientation) is usually in the neighborhood of $\pm 1^\circ$ [73, 117]. Additionally, the mask may not be perfectly aligned to the wafer flat. This alignment accuracy level sometimes reduces the precision of shapes fabricated from some processes. For example, the size of diaphragms formed after etching through a $500\mu m$ thick silicon wafer can vary by $50\mu m$ if the accuracy of the alignment is of the order of 1° [30, 66, 117]. In previous chapters, a Genetic Algorithm approach for MEMS mask-layout and process synthesis has been developed, where optimal mask-layout and process flow can be automatically generated for a given target shape. But in that approach, no variations, or noise factors, are considered during the design procedure: perfect mask alignment is assumed. In actual fabrication, the misalignment of the wafer flat, which implies the misalignment of mask-layout, is a noise factor which affects the quality of the fabricated device, and this variation of mask-layout misalignment can be taken as the uncontrollable variation when robust design is conducted. A robust design of mask-layout can synthesize mask-layouts that are robust to the misalignment. Figure 7.3 schematically demonstrates how to extend the Genetic Algorithm synthesis approach to achieve robust design by integrating noise factors into the synthesis iterations. By making the evaluation environment noisy, the searched optimal solution is robust to the variations. The implementation details and a robust design with S/N ratio for mask misalignment are described in the following.

Fitness Evaluation

In the previous Genetic Algorithm synthesis approach, the shape comparison result between the simulated shape and the target shape is the performance metric during the candidate evaluation, and the fitness values are calculated accordingly to ensure that candidate solutions with smaller shape mismatch values have higher fitness values. For robust design, as illustrated in section 7.3, to evaluate the performance of candidate solutions a proper performance statistic should be carefully chosen, which should combine the candidate solution's mean performance as well as robustness measure. When conducting the robust design (synthesis) of mask-layouts, the S/N ratio in the Taguchi method is used as the performance statistic. The mask-layout misalignment is considered as the uncontrollable noise with Gaussian distribution, and the scheme to calculate fitness values for all candidate solutions in a population is shown in Figure 7.6. For each candidate solution (mask-layout) in a generation, a series of randomly generated misalignment values are applied to the mask-layout before the simulation of the fabrication and shape comparison of the simulated shapes and the target shape are carried out. Function $f(M)$ in Figure 7.6 simulates the fabrication of mask M and calculates the shape mismatch value between the simulated shape and the target shape. A series of shape mismatch values will be produced for different misalignment values, and all the shape mismatch values are used to construct the S/N ratio which is taken as the performance statistic. This S/N ratio is the overall performance metric, and fitness values for all individuals in a population are calculated according to the S/N ratios to guide the search for an optimum design point. The optimal mask-layout found will have high robustness to mask misalignment noise.

Example

As an example, a robust synthesis was conducted which produces the optimal mask-layout, robust to mask misalignment, given a mesa as the target shape. The mask misalignment was considered to be a random variable with Gaussian distribution with $\mu = 0$ and $\sigma = 1°$. For each mask, 10 misalignment values were sampled and applied to the mask (*i.e.*, $n = 10$ in Figure 7.6), and performance statistic S/N ratio was calculated using the formula shown in Figure 7.6. The synthesized optimal mask-layout (the dark polygon) and the target shape (the mesa) are shown in Figure 7.7. To show the high robustness of this synthesized mask, the synthesis result was compared with the result from synthesis without considering mask misalignment (here called non-robust synthesis; see section 7.1).

Figure 7.8 shows the comparison of synthesis results from the robust synthesis and the non-robust synthesis. The first row shows the mask-layout from non-robust synthesis and the fabricated shapes when the mask misalignments are 0°, 1.5°, and 3°. The second row is for robust synthesis. Although both masks show a good match between the simulated shape and the target shape when

```
for each generation with genotypes $G_1, G_2, ..., G_N$:
{
    Decode each genotype $G_i$ to produce corresponding phenotype $P_i$;
    for(each phenotype $P_i$ (mask-layout))
    {
        $S/N = 0$;
        for($j = 0; j <= n; j++$)
        {
            Randomly generate mask misalignment noise $\Delta_j$
                according to distribution;
            $M_i = P_i$ rotated $\Delta_j$ degrees;
            Evaluate $ShapeMismatch_i = f(M_i)$;
            $S/N+ = ShapeMismatch_i/n$;
        }
        $S/N = -10\, log(S/N)$;
        $ShapeMismatch_i = S/N$;
    }
    Obtain fitness value for each $G_i$ according to $ShapeMismatch_i$ of $P_i$;
}
```

Figure 7.6: Schematic model of fitness evaluation

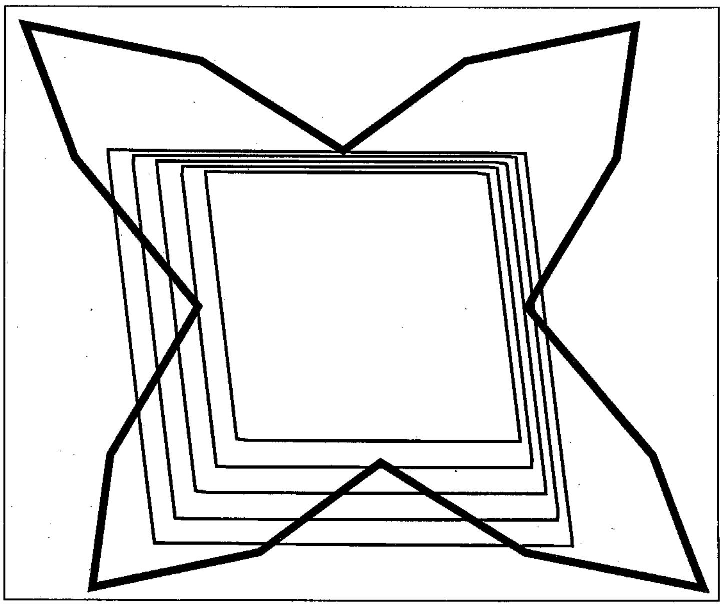


Figure 7.7: Synthesized result of a robust design

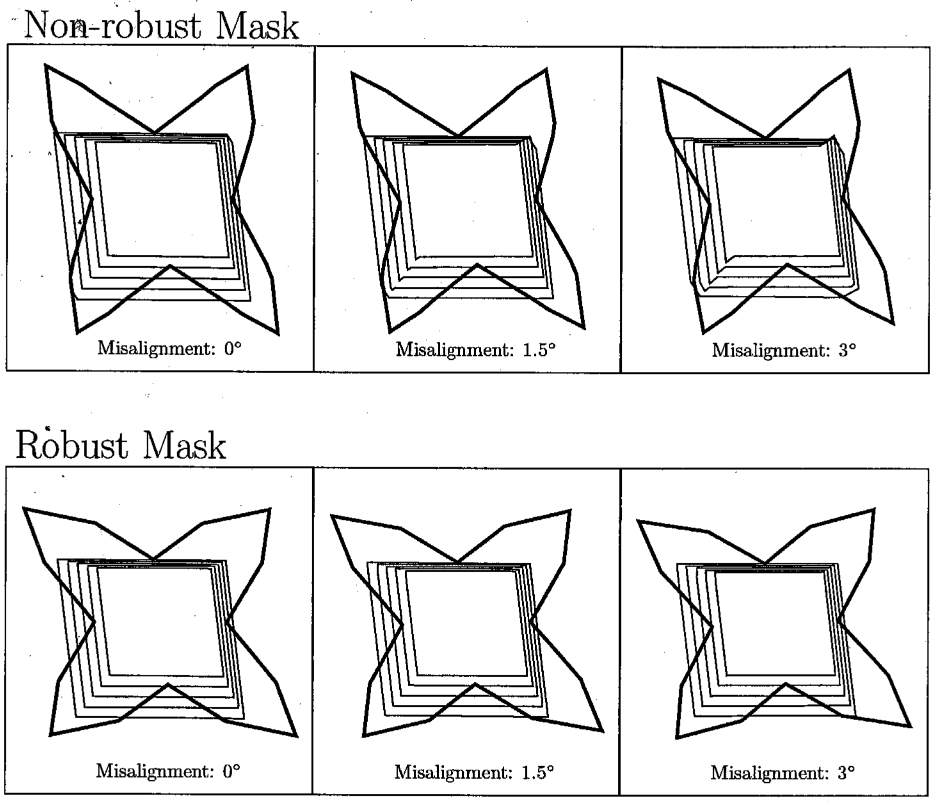


Figure 7.8: Comparison of robust synthesis and non-robust synthesis

the masks are perfectly aligned (0° misalignment, first column in Figure 7.8), the robust synthesis result exhibits much higher robustness. When the misalignment increases, the superiority of the mask-layout from robust synthesis can be easily observed: for a mask misalignment of 3°, the robust mask is still able to fabricate a 3-D shape fairly close to the target square mesa, while the mismatch between the shape resulted from the non-robust mask and the target shape is significant.

To statistically demonstrate that the result from robust synthesis does have higher robustness, both the robust mask and the non-robust mask were tested with a series (600 experiments) of randomly generated misalignment noises according to Gaussian distribution with $\mu = 0$ and $\sigma = 1°$. The shape mismatch values between the simulated shapes and the target shape obtained were used to construct the statistic mean (μ) and deviation (σ) using Equation (7.1) and Equation (7.2), and also S/N ratios for both cases using Equation (7.6). The values for μ, σ, and S/N ratios are shown in Table 7.1. Table 7.1 shows the robust synthesis result has smaller statistic mean (μ), smaller deviation (σ), and higher S/N ratio.

To visually show the superiority of the result from robust synthesis over non-robust synthesis,

Table 7.1: Statistics data for non-robust and robust synthesis results

	Non-Robust Synthesis Result	Robust Synthesis Result
μ	0.0385	0.0107
σ	0.0157	0.00443
S/N	27.63	38.72

using the shape mismatch values obtained from the 600 experiments for both masks, the histograms of the shape mismatch distribution for both masks under Gaussian distributed misalignment noise are plotted and shown in Figure 7.9 and Figure 7.10. Figure 7.9 is for the non-robust mask, and Figure 7.10 is for the robust mask. X axis is the shape mismatch value, and Y axis is the experiment frequency. The smaller mean and smaller deviation for the shape mismatch distribution for the robust synthesis result can be observed from the histograms. If, for example, acceptable devices have a shape mismatch (compared to the ideal target shape) of 0.025 or less, then nearly all devices fabricated with the robust mask-layout would be acceptable, while nearly 40% of the devices fabricated with the non-robust mask-layout have shape mismatch values larger than 0.025 and therefore would not be acceptable. Thus, in this case the robust mask-layout increases the yield from 60% to almost 100%.

Simulations were performed for both masks with different misalignment values ranging from 0° to 3.0°. Plots of shape mismatch *vs.* mask misalignment value for both masks are shown in Figure 7.11. For the non-robust mask-layout, the shape mismatch value increases significantly when the mask misalignment increases, while the curve for the robust mask-layout is relatively flat. The mask-layout from robust synthesis is more insensitive to the mask misalignment variations.

There is an interesting phenomenon in Figure 7.11. Since the non-robust synthesis searches for masks with good performance when perfectly aligned with no consideration of robustness, while the robust synthesis tries to balance the good performance of the mask and high robustness in some range of misalignment, it is expected that the curve of shape mismatch *vs.* mask misalignment would be flatter for the robust mask, but the non-robust mask has smaller shape mismatch value when perfectly aligned (0° misalignment) than the robust mask. But interestingly, in the example just described, as shown in Figure 7.11, although the search effort is the same for both the robust

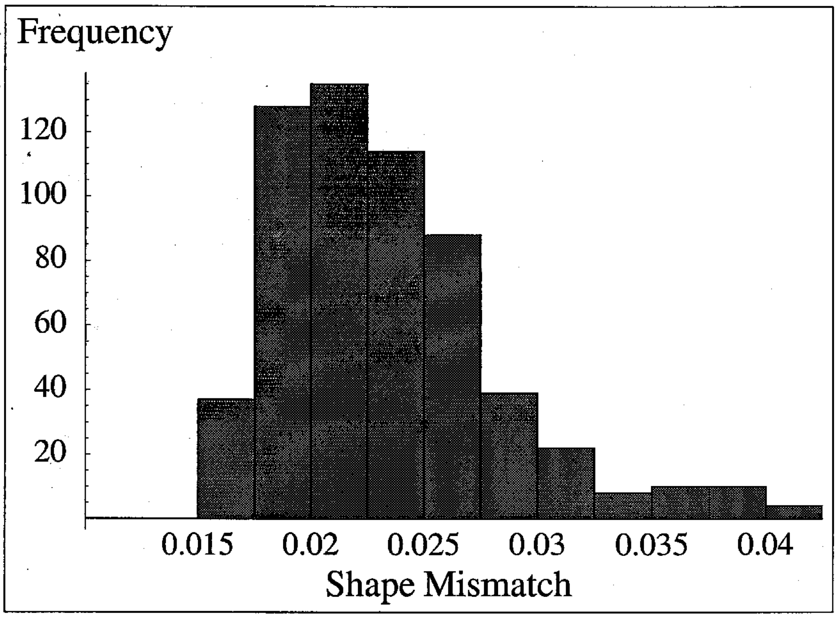


Figure 7.9: Histogram of fitness distribution for non-robust synthesis result

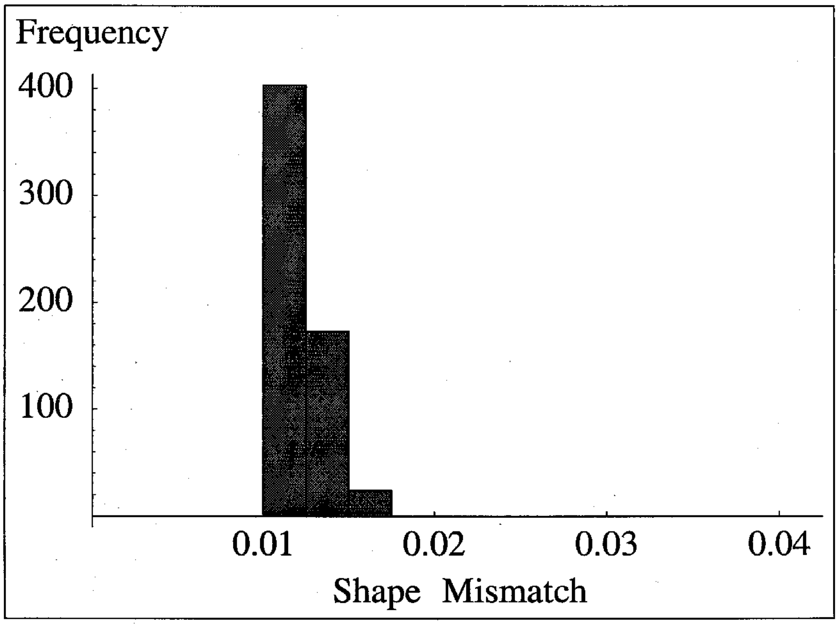


Figure 7.10: Histogram of fitness distribution for robust synthesis result

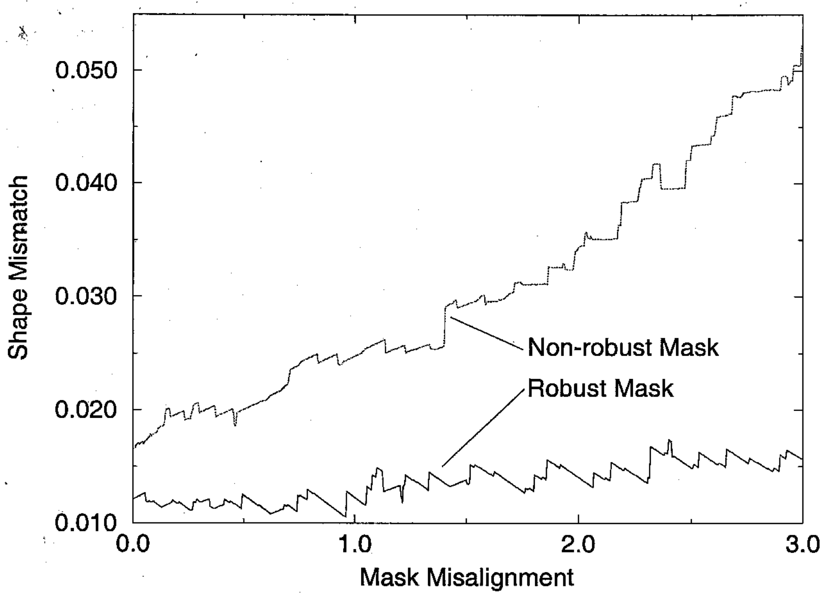


Figure 7.11: Shape mismatch *vs.* mask misalignment for both masks

synthesis and the non-robust synthesis (same iteration number, population size, *etc.*), the robust synthesis result shows better performance (smaller shape mismatch) than the non-robust synthesis result for any misalignment value, even when perfectly aligned (0° misalignment).

Computational Cost

Using the preceding approach, to calculate the performance statistic S/N ratio, each mask-layout needs to be evaluated for a series of samplings of Gaussian distributed misalignment noise. For each sampling, simulation of the fabrication and the shape comparison need to be performed once. If the sampling number is n, n simulations and n shape comparisons need to be performed to calculate the S/N ratio for each mask-layout. Recall that for the previous non-robust synthesis, for each mask-layout, only one simulation and one shape comparison need to be performed. Obviously the robust synthesis increases computational cost dramatically. Things get worse when there are more noise sources in the robust design. If n samplings are done for each noise, m noise sources require n^m samplings and therefore n^m simulations and n^m shape comparisons to calculate the performance statistic for each candidate solution. To overcome the dramatic computational cost increase, another scheme for robust design sampling and evaluation is proposed and tested. This scheme is described below.

7.5 Genetic Algorithms for Robust Design

To extend the applications of GAs to domains that have a noisy environment and require detection of robust solutions, schemes combining GAs and robustness evaluation techniques and robust solution searching techniques have been introduced [36, 40, 81, 116]. GAs with noisy fitness functions and GAs exploring hyper-rectangle design regions instead of design points in traditional GAs have been studied. For the robust mask-layout and process synthesis problem, a robust design scheme similar to the one introduced by Tsutsui [116], called GAs with a Robust Solution Searching Scheme (GA/RS^3), is applied, which is outlined below.

7.5.1 GAs with a Robust Solution Searching Scheme

In the following, GAs with a Robust Solution Searching Scheme (GA/RS^3) [116] is described for optimization problems where perturbations (noise) exist in the design parameters (solutions). In GA/RS^3, for each individual (candidate solution) in the population, perturbations are added to the phenotypic parameter values while evaluating the fitness of the individual. In other words, the individual's performance is evaluated in a "noisy environment." This may sound like the same as the previous robust design approach using S/N ratio. But actually the sampling and evaluation schemes are different. For robust design with S/N ratio, each candidate solution is evaluated for multiple samplings of the noise and all the evaluation results are used to construct the S/N ratio, and the S/N ratio is used to calculate the fitness value. If one candidate survives the competition and appears in the next generation, its S/N ratio does not need to be re-evaluated. For GA/RS^3, in one population, each candidate solution is evaluated for <u>one</u> sampling of the noise and the evaluation result is used to calculate the fitness value for this generation. Even if a candidate solution survives into the next generation, to calculate its fitness value in the next generation, it will be re-evaluated again for another random sampling of the noise. So for the robust design with S/N ratio, the robustness of a solution is captured by the S/N ratio, while for GA/RS^3, multiple evaluation of the solutions in a noisy environment ensures the final solution is robust. In noisy environments, individuals having "good" genotypic material would become extinct if they were highly sensitive to perturbations of phenotypic features, and individuals robust to these perturbations would have a better chance to survive the competition. By evaluating individuals in a noisy environment, individuals with higher robustness are favored. The details of the performance evaluation in a noisy environment are illustrated below.

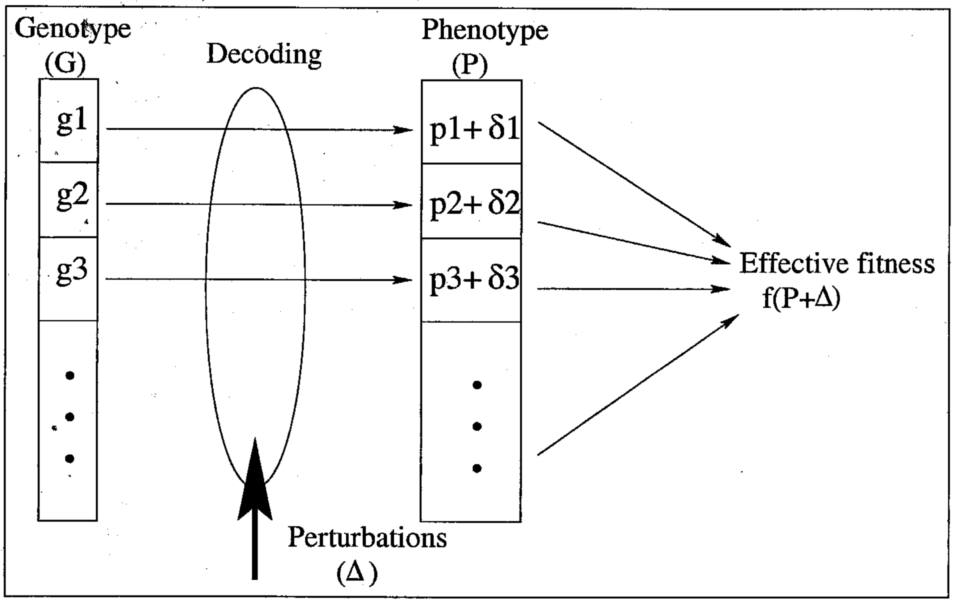


Figure 7.12: Fitness evaluation model in GA/RS^3

In a GA, suppose $G = (g_1, g_2, \ldots, g_m)$ is a genotypic string for an individual, $P = (p_1, p_2, \ldots, p_m)$ is the corresponding phenotypic parameter vector, and $f(P)$ the evaluation function for fitness. Instead of calculating the fitness value of the individual as $f(P)$, an evaluation function of the form $f(P + \Delta)$ is used, where $\Delta = (\delta_1, \delta_2, \ldots, \delta_m)$ is a random noise vector (see Figure 7.12). For each evaluation, the noise vector Δ is randomly generated according to its distribution probability. Because the individuals are evaluated in a noisy environment with presence of the noise, only individuals with both good and consistent (robust) performance can survive the competition. An individual with superb performance in one test but very sensitive to the noise will be eliminated through multiple random tests. The solutions thus determined are expected to be more robust against perturbations or noise. It should be noted that adding noise in the form $f(P + \Delta)$ may appear to be a mutation operation on a real-valued coding, but actually it is operationally different from mutation, since the noise is added only to the phenotype and it does not have any direct effect on individual genotypes. The perturbations are used only for judging the quality of a solution and for selection, and the genotypic string of the solution is unchanged.

Note that the preceding describes GA/RS^3 for cases where noises and perturbations are added to the design parameters directly. For example, in the mask-layout synthesis problem, the mask misalignment noise affects the design parameters (mask-layouts) directly by changing the orientations of the mask-layouts. The algorithm can also be applied to cases where noises exist not in the design parameters, but in the evaluation environment. For example, for the mask-layout synthesis problem, the etch rate data assumed in the synthesis may not be accurate. The actual etch rate variations is a noise existing in the fabrication environment. Although the design parameters (mask-layouts) are not affected by the etch rate variations, the performance of the mask-layouts are affected. GA/RS^3 can be applied to design mask-layouts robust to environment noises such as etch rate variations.

Now consider the computational time of the two different schemes for robust design: robust design with S/N ratio and GA/RS^3. As introduced before, Genetic Algorithms work in iterations. In each iteration (generation), there is a population of candidate solutions (called parent population). A new population of candidate solutions (called offspring population) is generated through genetic operations, and then elitist selection will select from both parent population and offspring population to create the parent population for next generation. For a Genetic Algorithm with iteration number M and population size N, if for each generation each candidate solution needs to be evaluated once, there are N evaluations for each generation. For the Genetic Algorithm robust design with S/N ratio, each new candidate solution (in the offspring population) needs to be evaluated n^m times to calculate the S/N ratio if there are m noise sources and n samplings are required for each noise source, while the candidate solutions in the parent population do not need to be re-evaluated because their S/N ratios are already known. The total number of evaluations for the Genetic Algorithm will be MNn^m. For GA/RS^3, in each generation, both the parent population and the offspring population need to be evaluated, which is $2N$ evaluations. The total number of evaluations for the Genetic Algorithm will be $2MN$.

Therefore, GA/RS^3 saves computational time compared with the Genetic Algorithm robust design with the S/N ratio. Genetic Algorithm Robust design with the S/N ratio evaluated the robustness of each candidate solution by explicit multiple samplings of the noise, while for GA/RS^3, the candidate solutions are evaluated in noisy environment while the iteration proceeds. In GA/RS^3, if one candidate solution is preserved from the initial (first) generation to the final generation, it is evaluated with random noise samplings N (iteration number) times. But for the newly generated candidate solution in the final generation, it is only evaluated once. GA/RS^3 distributes the sampling effort intelligently by utilizing the intrinsic exploration and exploitation power of the genetic operations, such that efficient evaluation of the robustness of the candidate solutions can be achieved for a limited number of noise samplings [116].

The GA/RS^3 approach to robust design will be applied to the mask-layout synthesis problem in the next section, and robust mask-layout synthesis is conducted for etch rate variations.

Orientation	Etching Rate (μm/hr)
100	25
110	25
311	30
111	5

Table 7.2: Etching rate data

7.6 GA/RS^3 Robust Design for Etch Rate Variations

The Genetic Algorithm approach to the mask-layout and process synthesis problem works by utilizing a process simulation. For the process simulation to predict fabrication results correctly, accurate etch rate data are needed. But generally, it's not easy to obtain accurate etch rate data. Etch rates for an etchant change with factors such as temperature, concentration, stirring, *etc.*, and they may even change during the etching process. The etch rate variations (or the inaccuracy in the etch rate model) present a problem to the Genetic Algorithm mask-layout synthesis approach: the synthesized optimal mask-layouts based on assumed etch rate data may not generate the desired shape in actual fabrication because the actual etch rates may be different. The GA/RS^3 approach to robust design is applied here to synthesize mask-layouts robust to the etch rate variations.

Test 1

As the first example, a robust synthesis was conducted to produce the optimal mask-layout robust to etch rate variations, given a mesa as the target shape. For the wet etching simulation program SEGS, the etch rate model assumed etch rate data as shown in Table 7.2. The actual etch rate in <311> direction during the fabrication was considered to be a Gaussian distributed random variable. The mean (μ) of the Gaussian distribution is the same as the <311> direction etch rate assumed in the etch rate model (30 μm/hr), and the standard deviation (σ) is 5% of the mean ($5\% \times 30 = 1.5$ μm/hr). The GA/RS^3 algorithm was implemented such that in each generation, the etching process is simulated for each mask-layout using a randomly generated <311> direction etch rate according to the Gaussian distribution with $\mu = 30$ and $\sigma = 1.5$. The synthesized optimal mask-layout (the dark polygon) and the target shape (the mesa) are shown in Figure 7.13. To show the high robustness of this synthesized mask, the synthesis result was compared with the result from synthesis without

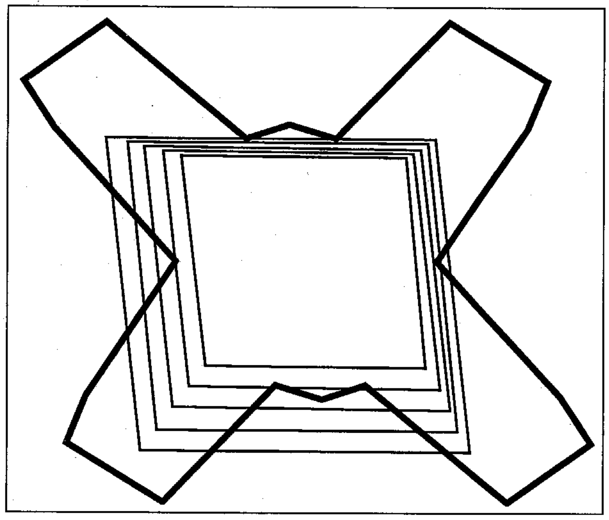


Figure 7.13: Synthesized result of a robust design

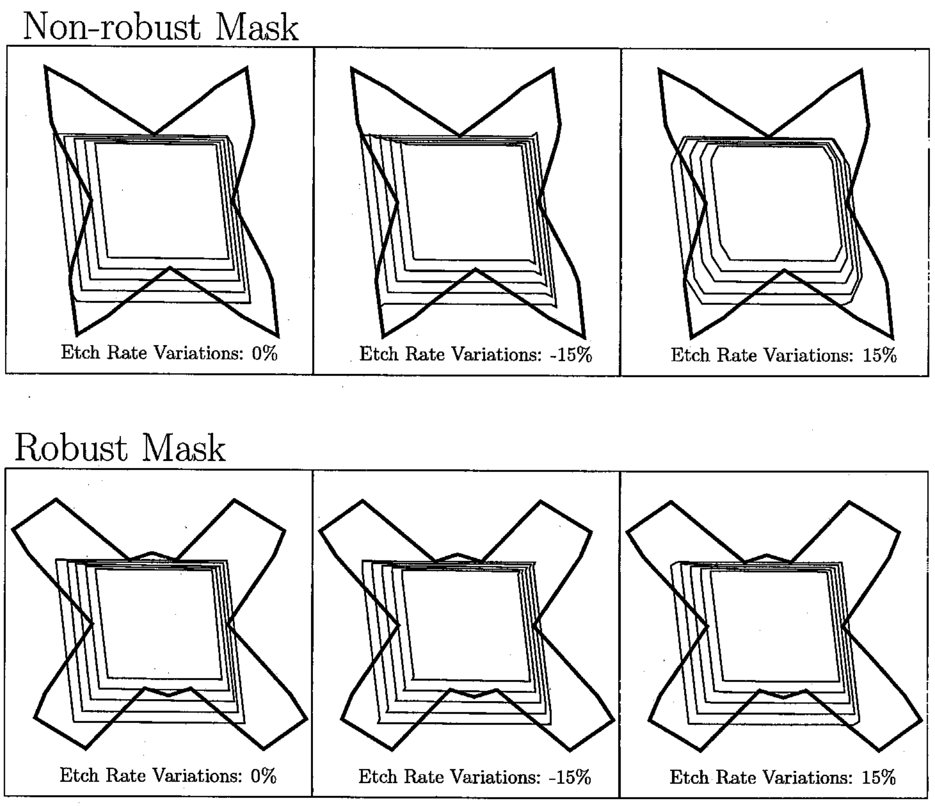

Non-robust Mask
Etch Rate Variations: 0%
Etch Rate Variations: -15%
Etch Rate Variations: 15%
Robust Mask
Etch Rate Variations: 0%
Etch Rate Variations: -15%
Etch Rate Variations: 15%

Figure 7.14: Comparison of robust synthesis and non-robust synthesis

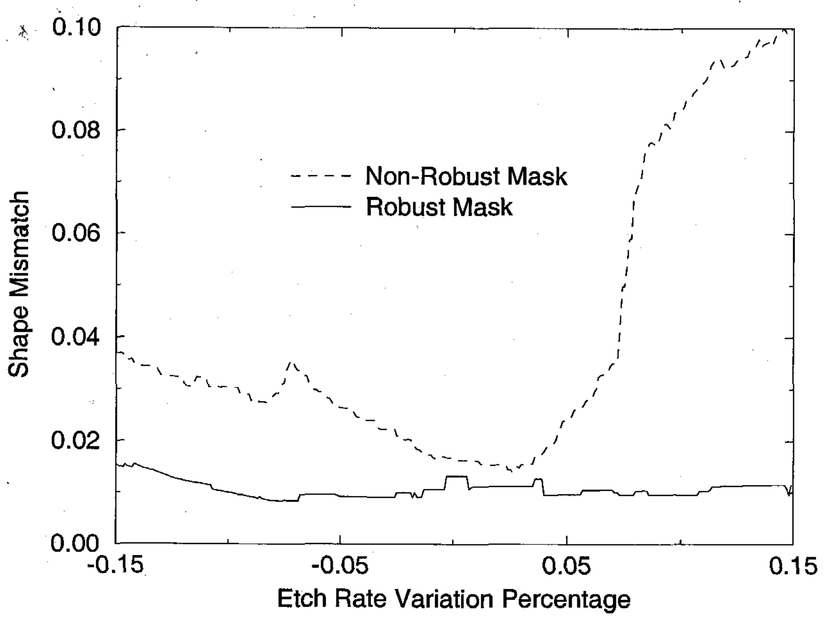


Figure 7.15: Shape mismatch *vs.* etch rate variations for both masks

considering the etch rate variations (here called non-robust synthesis).

Figure 7.14 shows the comparison of the synthesis results from the robust synthesis and the non-robust synthesis. The first row shows the mask-layout from non-robust synthesis and the fabricated shapes when the actual <311> direction etch rate used in the fabrication is 30 μm/hr, 25.5 μm/hr and 34.5 μm/hr (*i.e.*, the same as, 15% smaller than, and 15% larger than the <311> direction etch rate assumed in the etch rate model). The second row is for the robust synthesis. Although both masks show a good match between the simulated shape and the target shape when the actual <311> direction etch rate and the assumed etch rate in the etch rate model are the same (30 μm/hr), the robust synthesis result exhibits much higher robustness. When the actual <311> direction etch rate in the fabrication is 34.5 μm/hr, 15% larger than the <311> direction etch rate assumed in the etch rate model, the robust mask still generates a shape fairly close to the target mesa, while the mismatch between the fabricated shape from the non-robust mask and the target mesa is significant.

Plots of shape mismatch *vs.* <311> direction etch rate variations (the percentage of the difference of etch rate in <311> direction in the fabrication and the etch rate assumed in the etch rate model) for both masks were obtained by simulating the fabrication for both masks with different <311> direction etch rate values ranging from 15% smaller than the assumed <311> direction etch rate in the etch rate model to 15% larger than the assumed <311> direction etch rate. The plots for both masks are shown in Figure 7.15. The non-robust mask is more sensitive to the <311> direc-

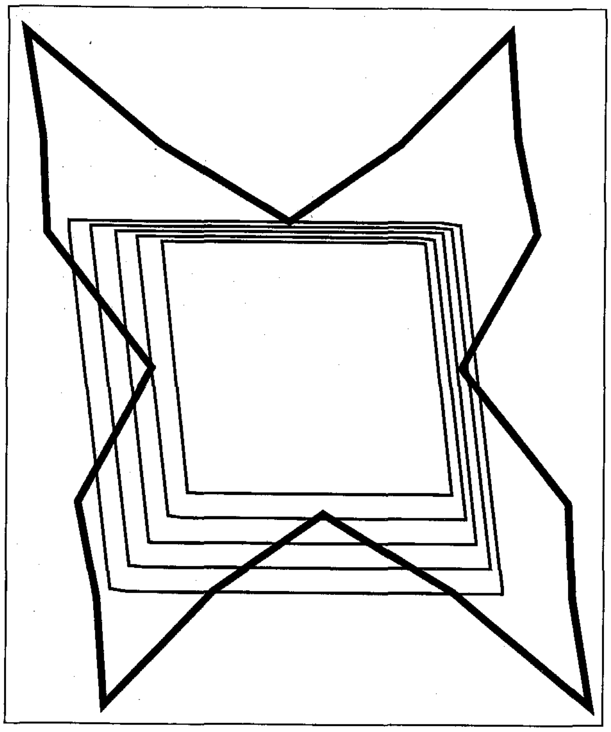
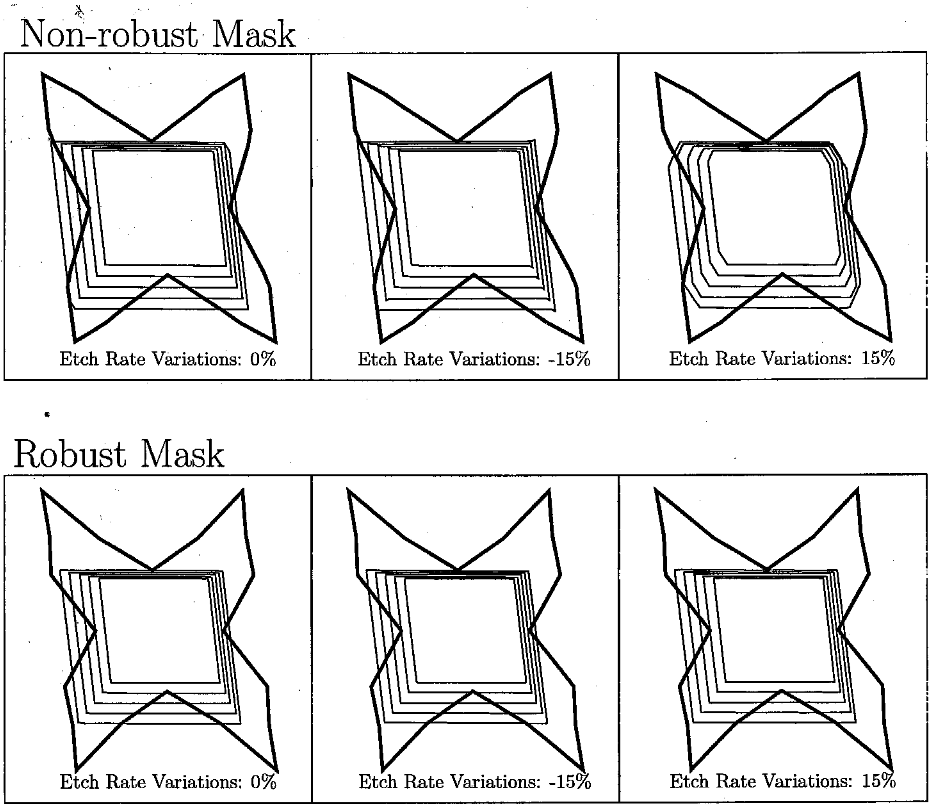


Figure 7.16: Synthesized result of a robust design

tion etch rate variations, and the shape mismatch value between the fabricated shape and the target shape increases significantly when the amount of the <311> direction etch rate variation increases, while the mask from robust synthesis is less sensitive to the etch rate variations.

Test 2

In the second test for robust mask-layout synthesis with etch rate variations, the same target shape is used, and the same etch rates as shown in Table 7.2 are assumed in the etch rate model. For etch rate variations, the etch rates in <110>, <311> and <111> directions are considered to be Gaussian distributed random variables. The means (μ) of the Gaussian distribution are the corresponding etch rates in these directions in the etch rate model (25 μm/hr, 30 μm/hr and 5 μm/hr), and the standard deviations (σ) are 5% of the means (1.25 μm/hr, 1.5 μm/hr and 0.25 μm/hr). The GA/RS^3 algorithm is implemented such that in each generation, the etching process is simulated for each mask-layout using randomly generated etch rates in <110>, <311> and <111> directions according to the Gaussian distribution. The synthesized optimal mask-layout (the dark polygon) and the target shape (the mesa) are shown in Figure 7.16. To show the high robustness of this synthesized mask, the synthesis result is compared with the result from non-robust synthesis



Non-robust Mask
Etch Rate Variations: 0%
Etch Rate Variations: -15%
Etch Rate Variations: 15%
Robust Mask
Etch Rate Variations: 0%
Etch Rate Variations: -15%
Etch Rate Variations: 15%

Figure 7.17: Comparison of robust synthesis and non-robust synthesis

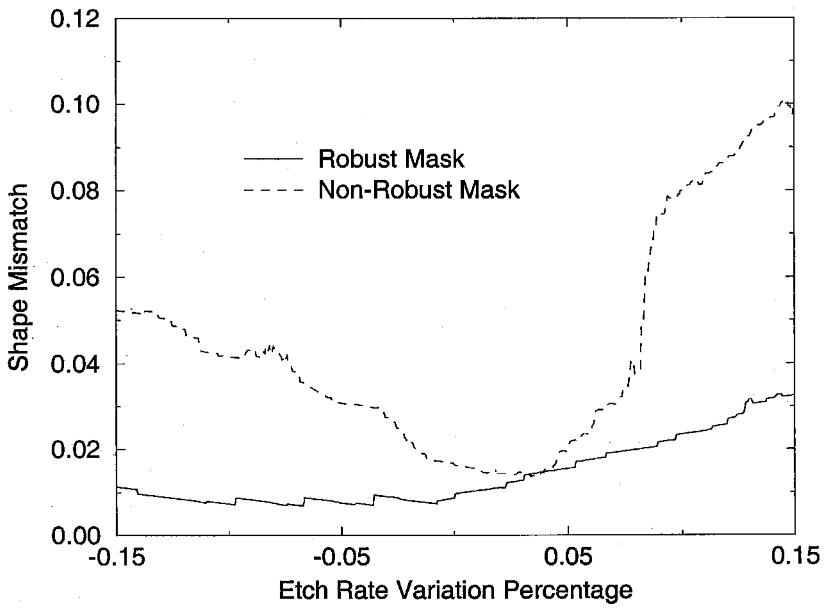


Figure 7.18: Shape mismatch *vs.* etch rate variations for both masks

without considering the etch rate variations.

Figure 7.17 shows the comparison of the synthesis results from the robust synthesis and the non-robust synthesis. The first row shows the mask-layout from non-robust synthesis and the fabricated shapes when the actual etch rates in <110>, <311> and <111> directions in the fabrication are the same as, 15% smaller than, and 15% larger than the ones in the etch rate model (25 μm/hr, 30 μm/hr and 5 μm/hr). The second row is for the robust synthesis. Although both masks show good match between the simulated shape and the target shape when the actual etch rates in <110>, <311> and <111> directions and the assumed etch rates in the etch rate model are the same, the robust synthesis result exhibits much lower sensitivity. When the actual etch rates in <110>, <311> and <111> directions are 15% larger than the assumed etch rates in the etch rate model, the robust mask still generates a shape fairly close to the target mesa, while the mismatch between the fabricated shape from the non-robust mask and the target mesa is significant.

Plots of shape mismatch *vs.* the etch rate variations (the percentage of the difference of etch rates in <110>, <311> and <111> directions in the fabrication and the etch rates assumed in the etch rate model) for both masks are obtained by simulating the fabrication for both masks with different etch rates in <110>, <311> and <111> directions ranging from 15% smaller than the assumed etch rates in the etch rate model to 15% larger than the assumed etch rates. The plots for both masks are shown in Figure 7.18. The non-robust mask is more sensitive to the etch rate variations, while the mask from robust synthesis is less sensitive.

7.7 Summary

This chapter focused on robust design. Since noise (variations) always exists in the fabrication procedure, the mask-layouts and process flows synthesized by the previous GA approach may generate unsatisfactory results if the mask-layouts and process flows are sensitive to the variations. Design solutions robust to the variations can generate satisfactory results under a variety of fabrication conditions and are highly desired. The previous Genetic Algorithm approach was modified to achieve robust mask-layout synthesis. By incorporating expected variations into the GA iterations, the algorithm is capable of producing solutions robust to the variations. Two approaches for robust design of mask-layouts and processes were developed. Genetic Algorithm robust design with the S/N ratio evaluates each candidate solution with multiple samplings of the noise, and the S/N ratio in the Taguchi method is constructed to guide the evolution. In GAs with a Robust Solution Searching

Scheme (GA/RS^3), the candidate solutions are evaluated in noisy environment while the iteration proceeds. These two approaches were tested for robust mask-layout synthesis with mask misalignment and etch rate variations, and mask-layouts robust to the variations have be synthesized. The high robustness of the synthesized mask-layouts was verified by comparing them with the mask-layouts synthesized from the previous non-robust synthesis approach.

Chapter 8

Conclusion

8.1 Summary

In this thesis, a Genetic Algorithm approach for the mask-layout and process flow synthesis problem has been developed. For a given desired target shape, an optimal mask-layout and process flow can be automatically generated using the Genetic Algorithm synthesis approach. The Genetic Algorithm manipulates and evolves a population of candidate solutions (mask-layouts and process parameters) by utilizing a process simulation tool to evaluate the performance of the candidate solutions. The process simulation tool is used to simulate the fabrication for each mask-layout with associated process flow to generate the 3-D device geometry. A shape matching algorithm compares the simulated device shapes with the desired target shape, and assigns a fitness value for each candidate solution in a population. For the mask-layout and process flow synthesis problem, encoding schemes, selection schemes, and genetic operations have been developed to effectively explore the solution space and control the evolution and convergence of the solutions.

The synthesis approach was tested for mask-layout and process synthesis for bulk wet etching. By integrating a bulk wet etching simulation tool into the Genetic Algorithm iterations, the algorithm can automatically generate proper mask-layout and process flow which can fabricate 3-D geometry close to the desired 3-D target shape. For structures with convex corners, complex compensation structures can be synthesized by the algorithm. More importantly, the process flow can also be synthesized. For multi-step wet etching processes, proper etchant sequence and etch times for each etch step can be synthesized automatically by the algorithm. When the choice of different process flows exists, the enlarged solution space makes the design problem more challenging. The ability to synthesize process flows makes the automatic design method more complete and more valuable.

The algorithm was further extended to achieve robust design. Since fabrication variations and modeling inaccuracy always exist, the synthesized solutions without considering these variations may not generate satisfactory results in actual fabrication. Robust designs using the Taguchi method and Genetic Algorithms with a Robust Solution Searching Scheme (GA/RS^3) have been developed to synthesize robust mask-layouts and process flows in "noisy" environment. Since the synthesis procedure considers the effect of variations in the fabrication procedures, the final synthesized solution will have high robustness to the variations, and will generate satisfactory results under a variety of fabrication conditions. The robust design approaches were implemented and tested for robust mask-layout design for mask misalignment and etch rate variations. Mask-layouts robust to mask misalignment noise and etch rate variations during the fabrication were synthesized. A typical robust synthesis run using GA/RS^3 took about 1 hour, and the synthesized mask-layout generally improved the yield significantly by exhibiting consistent performance under a variety of fabrication conditions.

8.2 Future Work

Further work can be done to expand the potential of the Genetic Algorithm approach to the mask-layout and process flow synthesis problem.

First, work can be done about the population size control for the Genetic Algorithm. In the algorithm, the user needs to specify the population size before running the synthesis. To ensure a reasonable amount of sampling of the solution space, the population size cannot be too small. But setting a too large population size dramatically increases the computation time, especially when the process simulation is time-consuming. To set an appropriate population size, trials and tunings are needed and trade-off between the sampling of the solution space and the computation cost needs to be made. A possible scheme is to use variable population size. The population size is set large when starting the synthesis to hold enough samplings of the solution space, and when the evolution converges, the population size is decreased to eliminate duplicate or similar candidate solutions and save computation cost.

Second, the algorithm can be extended by integrating other process simulation tools into the Genetic Algorithm iterations. As described earlier, the Genetic Algorithm approach works by utilizing a process simulation tool as performance evaluation, and the bulk wet etching simulator SEGS is used in the examples shown to synthesize mask-layout and process for wet etching steps. SEGS can only simulate wet etching steps, while in actual fabrication, etching sometimes is used together with "adding" techniques such as deposition. More complex structures can be generated by using different fabrication techniques together, and the design for mask-layouts and process flows is more challenging because of the enlarged solution space. The potential of the Genetic Algorithm approach to the mask-layout and process synthesis problem can be fully realized by utilizing a more complete process simulation tool. Of course, for a particular fabrication technique and related applications, ways to represent and compare the geometry need to be developed.

Third, the approach can be customized in the sense that other performance evaluation criteria can be used in addition to shape matching. Shape matching between the simulated shapes and the target shape was used as the performance evaluation criteria in the test examples, and candidate solutions with close shape match were assigned with high fitness values. In the robust design with Taguchi method, the robustness measure S/N ratio, which is a combination of shape mismatch mean and deviation, was used as the performance evaluation criteria, and candidate solutions are considered to be fit when they have not only a small shape mismatch mean, but also a small deviation. The algorithm can be customized to use other performance evaluation criteria so that the final synthesized solution will exhibit a desired design function. For example, to achieve a material-efficient mask-layout design, mask-layout with smaller area is considered better than mask-layout with bigger area. The area of the mask-layout can be used as one of the performance evaluation criteria, and mask-layout with small area can be synthesized. Figure 8.1 shows a synthesized mask with small y-direction dimension. To fabricate an array of square mesas with high density in the y-direction, mask-layout with small y-direction dimension is needed. During the mask-layout synthesis, for each candidate mask, its y-direction dimension is measured and combined with the shape mismatch value between the simulated shape and the target shape to calculate the fitness value for the mask. The fitness value calculation formula ensures that a mask with small y-direction dimension and small shape mismatch value will be assigned with a high fitness value. The final synthesized mask will not only fabricate a geometry close to the target shape, but also have a small y-direction dimension.

Finally, as illustrated in Chapter 1, the mask-layout and process synthesis addresses only part of the design problem in the synthesis and optimization approach: to synthesize mask-layouts and process flows for a given device shape. Issues about Geometry synthesis (synthesizing device shape from a device function specification), and integration of mask-layout and process synthesis and geometry synthesis have not been examined. The principles of the Genetic Algorithm iterative

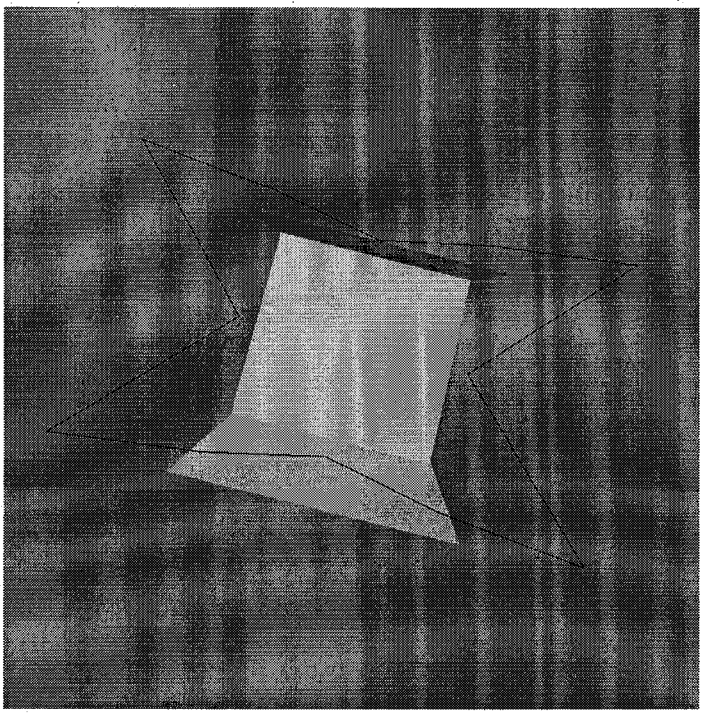


Figure 8.1: A synthesized mask-layout with small y-direction dimension

synthesis approach developed here can be borrowed to develop schemes for geometry synthesis. By utilizing process synthesis and geometry synthesis, for a specified device function, the device shape and mask-layout and process to produce the device can be automatically generated, and designers can fully focus on the device functional design.

Appendix A

Exploitation and Exploration

Most classical optimization methods find the optimal solution by determining a sequence of steps leading to the optimum based on the gradient or higher order derivatives of the objective function. When such information is unavailable, search becomes the only practical method. Search can be performed by either *blind strategies* or *heuristic strategies* [12]. Blind search strategies do not use information about the problem domain. Heuristic search strategies use additional information to guide the search along with the best search directions. These two strategies focus on two different but equally important aspects during the search/optimization: exploiting the best solution and exploring the search space [13]. Exploration investigates new and unknown areas in the search space, and exploitation makes use of knowledge found at points previously visited to help find better points. Hill-climbing is an example of a strategy which exploits the best solution for possible improvement while ignoring the exploration of the search space. Random search is an example of a strategy which explores the solution space while ignoring the exploitation of the promising regions of the search space. These two strategies are contradictory, and a good search algorithm must find a tradeoff between the two. Genetic algorithms are a class of general-purpose search methods combining elements of directed and stochastic search which can make a remarkable balance between exploration and exploitation of the search space. At the beginning of genetic search, there is a widely random and diverse population and crossover operator tends to perform widespread search for exploring all solution space. As the high fitness solutions develop, the crossover operator provides exploitation in the neighborhood of each of them. In other words, what kinds of searches (exploitation or exploration) a crossover performs are determined by the environment of the genetic system (the diversity of population).

Appendix B

Fitness Scaling

If raw fitness values are used for fitness proportionate selection, without any scaling or normalization, then one of two things can happen. If the fitness range is too large, then only a few good individuals will be selected. This will tend to fill the entire population with similar chromosomes and will limit the ability of the GA to explore the search space. On the other hand, if the fitness values are too close to each other, then the GA will tend to select one copy of each individual, with only random variations in selection. Consequently, it will not be guided by small fitness variations and will be reduced to random search. In early generations, there is a tendency for a few super chromosomes to dominate the selection process, and in later generations, when the population is largely converged, competition among chromosomes is less strong and a random search behavior will emerge. Fitness scaling is used to scale the raw fitness values so that the GA sees a reasonable amount of difference in the scaled fitness values of the best versus the worst individuals [77]. Thus, fitness scaling controls the selection pressure or discriminating power of the GA.

Beginning with [27], scaling of objective function values has become a widely accepted practice and several scaling mechanisms have been proposed. In general, the scaled fitness F'_k derived from the raw fitness F_k for chromosome k can be expressed as follows:

$$F'_k = G(F_k)$$

where the mapping function $G(\cdot)$ transforms the raw fitness into scaled fitness. The function $G(\cdot)$ may take different forms to yield different scaling methods, such as linear scaling, sigma truncation, power law scaling, *etc.* Some of them are illustrated briefly below. For detailed description, see [44].

Linear Scaling When the mapping function takes the form of a linear transformation, the following linear scaling method applies:

$$F'_k = a \times F_k + b$$

where parameters a and b are normally selected such that the average chromosome receives one offspring copy on average, and the best receives the specified number of copies (usually two). Linear scaling adjusts the fitness values of all chromosomes in such a way that the best chromosome gets a fixed number of expected offspring and thus prevents it from reproducing too many offspring. This method may give negative fitness values that are usually taken as zero.

Sigma Truncation For Sigma truncation [49],

$$F'_k = F_k - (\bar{F} - c \times \sigma)$$

where c is a small user-defined integer called the sigma scaling factor, σ is the standard deviation of the fitness of the population, and $\bar{F}$ is the average raw fitness value. Negative scaled fitnesses F'_k are set to zero.

This scales the fitness such that, if the raw fitness is $\pm k$ standard deviations from the population average, the fitness is

$$F'_k = (c \pm k)\sigma.$$

This means that any individual worse than c standard deviations from the population mean $(k = c)$ is not selected at all. The usual value of c reported in the literature is between 1 and 5.

Boltzmann Selection Boltzmann selection [79] is a nonlinear scaling method for proportionate selection, using the following scaling function:

$$F'_k = e^{F_k/T}$$

where T is a user-defined control parameter. The selection pressure can be adjusted by assigning T high or low.

Appendix C

Tournament Selection

Like fitness proportionate selection, tournament selection is a stochastic selection scheme. In binary tournament selection [51], two individuals are taken at random, and the better individual is selected from the two. If binary tournament selection is being done without replacement, then the two individuals are set aside for the next selection operation, and they are not replaced into the population. Since two individuals are removed from the population for every individual selected, and the population size remains constant from one generation to the next, the original population is restored after the new population is half-filled. Therefore, the best individual will be selected twice, and the worst individual will not be selected at all. The number of copies selected of any other individual cannot be predicted except that it is either zero, one or two. In binary tournament selection with replacement, the two individuals are immediately replaced into the population for the next selection operation.

Binary tournament selection was generalized to *tournament selection* which works by taking a random uniform sample of a certain size $q > 1$ from the population, selecting the best of these q individuals to survive for the next generation. This method has gained increasing popularity because it is easy to implement, computationally efficient, and allows for fine-tuning of selection pressure by increasing or decreasing the tournament size q.

Appendix D

Crossover

D.1 Binary Crossover

The “traditional” GA uses 1-point crossover, where the two mating chromosomes (binary strings) are each cut once at corresponding points, and the sections after the cuts exchanged. The two offspring each inherit some genes from each parent. See Figure D.1 for detail.

Many different binary crossover schemes have been devised, often involving more than one cut point. In 2-point crossover (and multiple-point crossover in general), chromosomes are regarded as loops formed by joining the ends together. To exchange a segment from one loop with that from another loop requires the selection of two cut points, as shown in Figure D.2. Obviously, 1-point crossover can be viewed as 2-point crossover with one of the cut points fixed at the start of the string, and it’s fair to say that 2-point crossover is more general than 1-point crossover although they both perform the same task (exchanging single segment of two parents). Researchers now agree that 2-point crossover is generally better than 1-point crossover. The effectiveness of

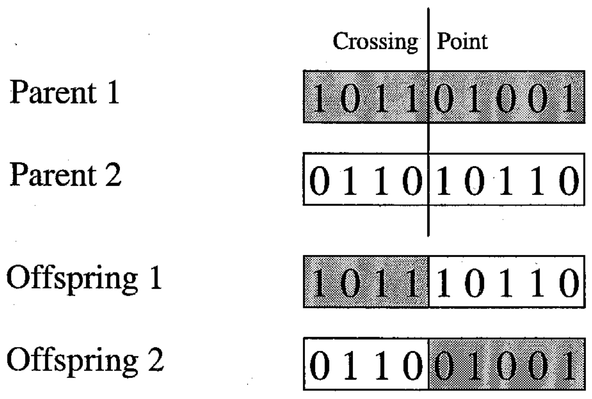


Figure D.1: One-point binary crossover

Chromosome: 1 1 0 1 1 0 0 1 0 1 0 0 1 1 0 1 0

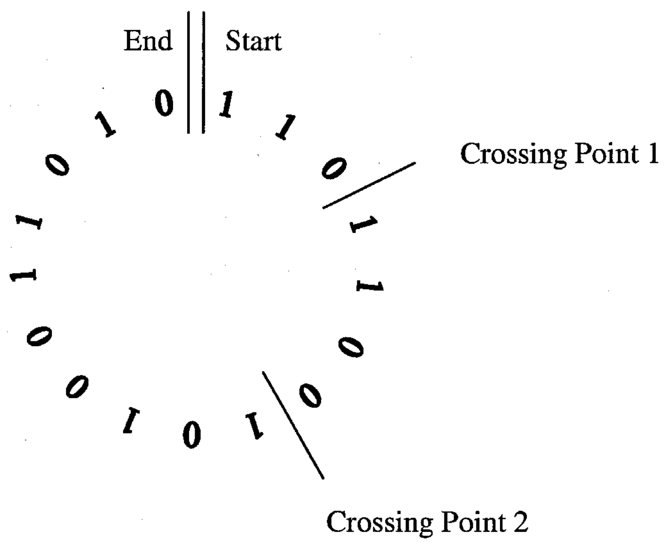


Figure D.2: Two-point binary crossover

multiple-point crossover was investigated in [27], and it was concluded that 2-point crossover gives the best performance, but that adding more crossover points reduces the performance of the GA. The problem with adding additional crossover points is that building blocks are more likely to be disrupted. However, an advantage of having more crossover points is that the solution space may be searched more thoroughly.

Uniform crossover is another binary crossover scheme which is radically different from 1-point crossover. In uniform crossover, after a pair of parents are selected, a crossover mask, which is also a binary string with the same length as the chromosomes, is randomly generated. An offspring is created by assigning each gene in the offspring by copying the corresponding gene from one or the other parent, chosen according to the crossover mask. Where there is a 1 in the crossover mask, the gene is copied from the first parent, and where there is a 0 in the mask, the gene is copied from the second parent, as shown in Figure D.3. The process is repeated to produce the second offspring. The offspring therefore contain a mixture of genes from each parent, and the number of effective crossing points is not fixed, but will average $L/2$ where L is the chromosome length.

A comparison of different binary crossover operators, including 1-point, 2-point, multi-point and uniform crossover, was undertaken in [32], both theoretically and empirically. It was found that none of them is the consistent winner, and there was not more than 20% difference in speed among the techniques.

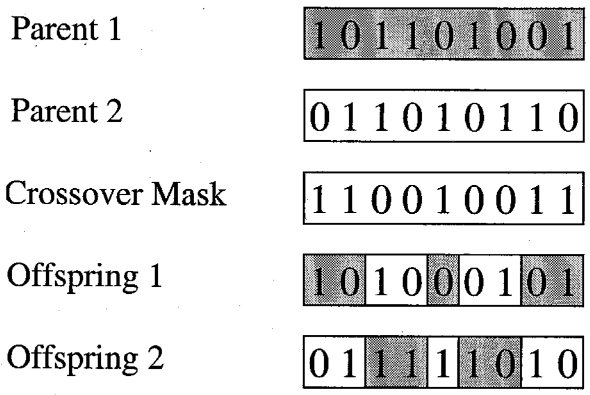


Figure D.3: Uniform binary crossover

D.2 Real Crossover

In real encoding implementations, each chromosome is encoded as a vector of real numbers with the same length as the solution vector. In recent years, several crossover schemes have been proposed for real number encoding, and most of them belong to the following two categories: conventional crossovers (including simple and random), and arithmetic crossovers.

The conventional crossovers are made by extending the canonical crossover operators for binary representation into the real coding case. The arithmetic crossovers are constructed by borrowing the concept of linear combination of vectors from the area of convex sets theory. These crossover operators are described in detail below.

D.2.1 Conventional Crossovers

Conventional crossovers include simple crossovers and random crossovers. Random crossovers have been introduced in section 2.6. Simple crossovers [110, 111] include 1-point, 2-point, multi-point and uniform crossover. These crossover operators are analogous to those of the binary implementation, except that each gene is now a real number rather than a binary bit (0 or 1).

D.2.2 Arithmetic Crossovers

Let x_1 and x_2 be two parent chromosomes (real vectors). Arithmetic crossovers are defined as the combination of two vectors as follows:

$$x_1' = \lambda_1 x_1 + \lambda_2 x_2$$

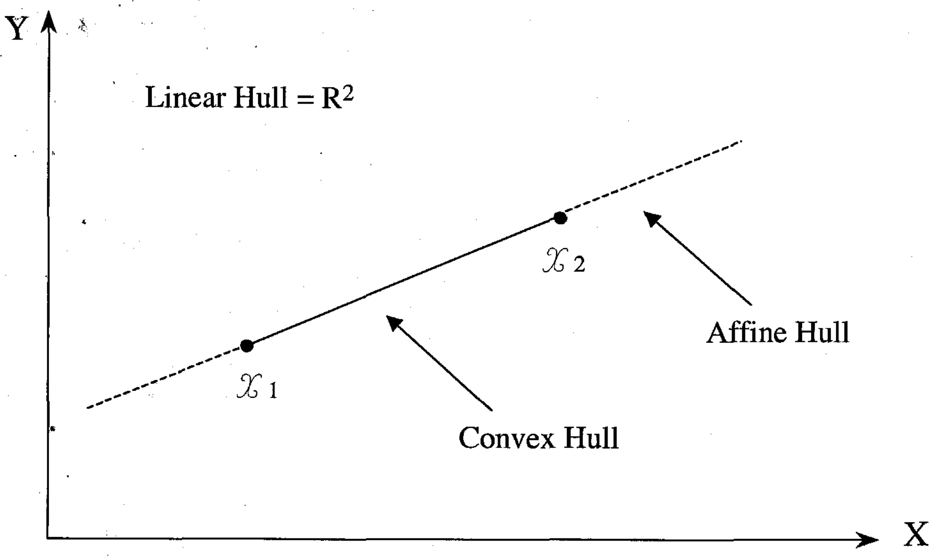


Figure D.4: Arithmetic crossover

$$x'_2 = \lambda_1 x_2 + \lambda_2 x_1$$

By restricting the coefficients λ_1 and λ_2 in different ways, the equations above produce three different kinds of arithmetic crossover [45, 119, 79]:

linear crossover	λ_1, λ_2 are real
affine crossover	$\lambda_1 + \lambda_2 = 1$
convex crossover	$\lambda_1 + \lambda_2 = 1, \quad \lambda_1 > 0, \lambda_2 > 0$

The names *linear*, *affine* and *convex* are borrowed from convex set theory, in which a combination of two vectors, x_1 and x_2, is called linear, affine, or convex when λ_1 and λ_2 are restricted to be real, $\lambda_1 + \lambda_2 = 1$, or $\lambda_1 + \lambda_2 = 1, \lambda_1 > 0, \lambda_2 > 0$. The convex crossover appears to be the most commonly used. When the restriction that $\lambda_1 = \lambda_2 = 0.5$ is applied, the special case of the averaging crossover is produced [26].

A geometric explanation of arithmetic operators for a two-dimensional case is shown in Figure D.4. x_1 and x_2 are two parent vectors. The offspring generated with convex crossover constitute the so-called convex hull. Similarly, the offspring generated with affine crossover constitute the affine hull and the offspring generated with linear crossover constitute the linear hull. In the two-dimensional case (Figure D.4), the solid line connecting the two parents is the convex hull, and the solid and dashed lines are the affine hull and the linear hull is the whole space.

Appendix E

Schema Theorem

Most research into GAs has so far concentrated on finding empirical rules for getting them to perform well. Exactly why Genetic Algorithms work is a subject of some controversy, with much more work being required before all questions can be finally answered. Nevertheless, several hypotheses have been put forward which can partially explain the success of GAs. This can be used to help us implement good GA applications.

Holland's schema theorem [53] was the first rigorous explanation of how GAs work, and it formed the basis of most theoretical work on the topic. The schema theorem explains the power of the GA in terms of how schemata are processed.

E.1 Schema

A schema (plural *schemata*) is a fixed template describing a subset of strings with similarities at certain defined positions. Thus, strings which contain the same schema contain, to some degree, similar information. Here only binary alphabets will be considered, allowing templates to be represented by the ternary alphabet {0, 1, #}. The meta-symbol # is called the "don't care" element, and within any string the presence of the meta-symbol # at a position implies that either a 0 or a 1 could be present at that position. A particular chromosome is said to be an instance of a particular schema, or contain a particular schema if it matches that schema, with the # symbol matching anything. So for example, 101010 and 010010 are both instances of the schema 1##010. Conversely, two examples of schemata that are contained within 101010 are 1010#0 and 1#10##. A schema is a partial solution and represents a set of possible fully specified solutions. A schema with m specified elements and $(n-m)$ #s can be considered to be an $(n-m)$ dimensional hyperplane in the solution space. All points on that hyperplane are instances of the schema.

In a typical binary-coded GA, where the chromosomes are binary strings, each string in the population is an instance of 2^L schemata (or say each string contains 2^L schemata), where L is the length of each individual string. Therefore, a population of N chromosomes could contain between 2^L and $N2^L$ possible schemata, since there may be duplications. The total number of different schemata contained in all possible strings is 3^L, since each gene in a schema may be 0, 1, or #. In general, for an alphabet of cardinality (or distinct characters) k, there are $(k+1)^L$ schemata. for a population of N chromosomes, there could be between k^L and Nk^L possible schemata. Thus, although there are only N chromosomes in the population, a much greater number of schemata are processed in parallel, and this property is called *implicit parallelism* [53].

A schema represents a region of the search space and the area of the search space represented by a schema and the location of this area depend on the number and location of the meta-symbols within the schema. The regions of the search space represented by schemata such as 1#### are much larger than schemata such as 1110#. Schemata are typically classified by their *defining length* and their *order*. The order o of a schema S is the number of positions within the schema that are not defined by a meta-symbol, *i.e.*,

$$o(S) = L - m$$

where m is the number of meta-symbols and L is the schema length. In other words, the order is the number of fixed positions within the schema:

$$S = \#1\#0\#; \qquad o(S) = 2$$

The defining length d specifies the distance between the first and last non meta-symbol characters within the schema:

$$S = \#1\#0\#; \qquad d(S) = 4 - 2 = 2$$

In general, low order schemata cover large regions of space and high order schemata cover much smaller regions, which is illustrated in Figure E.1. In Figure E.1, binary strings with 4 bits are used to represent integers in the range [0, 15]. Low order schema 1### covers half of the space ([8, 15]), and high order schema 01#0 covers a much smaller region (it actually represents only integers 4 and 6).

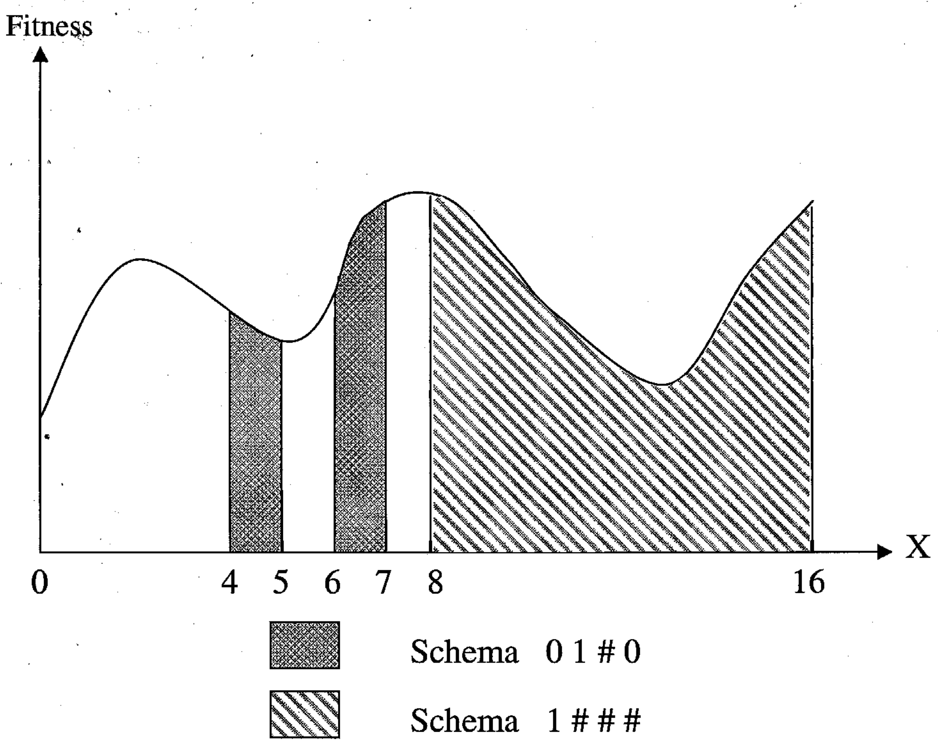


Figure E.1: Visualization of regions of schemata

E.2 Schema Processing

When genetic operators, such as selection, crossover and mutation, are applied to the population, the chromosomes are modified and the distribution of the schemata is changed accordingly. For any particular chromosome (string) within a GA, it can get fragmented by crossover, attacked by mutation or simply thrown away by the selection operator. Because the selection mechanism favors chromosomes with high fitness, fitter parents, which are expected to contain some good schemata, will produce more offspring. Therefore, the number of instances of good schemata tends to increase, and the number of instances of bad schemata tends to decrease. The change of the number of instances of a particular schema during a GA run can be roughly estimated and this estimation throws light on how GAs work.

The combined effect of selection, crossover, and mutation gives the so-called *reproductive schema growth equation*:

For a particular schema S, if $\Phi(S, g) > 0$ is the number of instances of S within the population at generation g,

$$\Phi(S, g+1) \geq \frac{f(S,g)}{\bar{f}(g)}\Phi(S,g)(1 - P_c\frac{d(S)}{L-1} - o(S)P_m)$$

where $f(S,g)$is the average fitness of all instances of S at generation g, and $\bar{f}(g)$ is the average fitness of generation g. P_c and P_m are the crossover probability and mutation probability, and $d(S)$ and o(S) are the defining length and order of S. The equation above indicates the expected number of strings matching a schema S in the next generation as a function of the actual number of strings matching the schema in the current generation, the relative fitness of the schema, and its defining length and order.

The final result of the growth equation can be stated as follows:

Schema Theorem: Short (defining length), low-order, above-average schemata receive exponentially increasing trials in subsequent generations of a Genetic Algorithm.

The short, low-order, above-average schemata are termed *building blocks* by Goldberg [49]. An immediate result of the schema theorem is that GAs explore the search space by building blocks which, subsequently, are combined into larger blocks through crossover. The *building block hypothesis* states that GAs attempt to find highly fit solutions to the problem under consideration by the juxtaposition of these building blocks:

Building Block Hypothesis: A Genetic Algorithm seeks near-optimal performance through the juxtaposition of short, low-order, high-performance schemata, called the building blocks.

Bibliography

[1] M.M. Abu-Zeid. Corner undercutting in anisotropically etched isolation contours. *Journal of the Electrochemical Society*, 131:2138–2142, 1984.

[2] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995.

[3] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. B. Mitchell. An efficiently conputable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, vol. 13, No. 3:209–215, 1991.

[4] Esther M. Arkin, L. Paul Chew, Daniel P. Huttenlocher, Klara Kedem, and Joseph S. B. Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:209–215, 1991.

[5] M. J. Atallah. A linear time algorithm for the hausdorff distance between convex polygons. *Information Processing Letter*, 17:207–209, 1983.

[6] T. Bäck. Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. In D. Fogel, editor, *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 57–62, New York, 1994. IEEE.

[7] T. Bäck and F. Hoffmeister. Extended selection mechanism in genetic algorithms. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 92–99, San Mateo, CA, 1991. Morgan Kaufmann, Publishers.

[8] Thomas Bäck, Ulrich Hammel, and Hans-Paul Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–15, April 1997.

[9] James E. Baker. Adaptive selection methods for genetic algorithms. In John Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, pages 101–111, Hillsdale, NJ, 1985. Lawrence Erlbaum Associates.

[10] James E. Baker. Reducing bias and inefficiency in the selection algorithm. In *The Second International Conference on Genetic Algorithms*, Hillsdale, NJ, 1987. Lawrence Erlbaum Associates.

[11] M. Bao, C. Burrer, J. Esteve, J. Bausells, and S. Marco. Etching front control of (110) strips for corner compensation. *Sensors and Actuators A-Physical*, 37-38:727–732, 1993.

[12] L. Bolc and J. Cytowski. *Search Methods for Artificial Intelligence*. Academic Press, London, 1992.

[13] L. Booker. Improving search in genetic algorithms. In *Genetic Algorithms and Simulated Annealing*. Pitman, New York, 1987.

[14] J. W. Boyse. Data structure for a solid modeller. In *NSF Workshop on the Representation of Three-Dimensional Objects*, 1979.

[15] Ernesto Bribiesca. Measuring 3-D shape similarity using progressive transformations. *Pattern Recognition*, vol. 29, No. 7, 1996.

[16] R. A. Buser, Selden B. Crary, and O. S. Juma. Integration of the anisotropic-silicon-etching program ASEP within the CAEMEMS CAD/CAE framework. In *Proceedings of Micro Electro Mechanical Systems (MEMS '92)*, pages 133–138, New York, February 1992. IEEE.

[17] R. A. Buser and N. F. de Rooij. ASEP: A CAD program for silicon anisotropic etching. *Sensors and Actuators A-Physical*, 28:71–78, 1991.

[18] H. Camon and A. Gue. Modelling of an anisotropic etching in silicon-based sensor applications. *Sensors and Actuators A-Physical*, 33:103–105, 1992.

[19] C.D. Chapman, K. Saitou, and M.J. Jakiela. Genetic algorithms as an approach to configuration and topology design. *ASME Journal of Mechanical Design*, 116(4):1005–1012, 1994.

[20] Wei Chen, J. K. Allen, Kwok-Leung Tsui, and F. Mistree. A procedure for robust design: Minimizing variations caused by noise factors and control factors. *ASME Journal of Mechanical Design*, 118:478–485, 1996.

[21] Wei Chen, Margaret Wiecek, and Jinhuan Zhang. Quality utility - a compromise programming approach to robust design. *ASME Journal of Mechanical Design*, 121(2):179–187, 1999.

[22] J. Cohoon and W. Paris. Genetic placement. *IEEE Transactions on Computer-Aided Design*, 6:1272–1277, 1987.

[23] David Corne and Peter Ross. Practical issues and recent advances in job- and open-shop scheduling. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 531–546. Springer-Verlag, Berlin, Germany, 1997.

[24] P. Cox, H. Maitre, M. Minoux, and C. C. Ribeiro. Optimal matching of convex polygons. *Pattern Recognition Letter*, 9:327–334, 1989.

[25] Selden B. Crary and Y. Zhang. CAEMEMS: An intergrated computer-aided engineering workbench for micro-electro mechanical systems. In *Proceedings of Micro Electro Mechanical Systems (MEMS '90)*, pages 113–115, New York, 1990. IEEE.

[26] L. Davis. *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold, 1991.

[27] Kenneth A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. thesis, The University of Michigan, Ann Arbor, MI, 1975.

[28] G. DeLapierre. Anisotropic crystal etching: A simulation program. *Sensors and Actuators A-Physical*, 31:267–274, 1992.

[29] D. Dietrich and J. Fruhauf. Computer simulations of the development of dish-shaped deepenings by orientation-dependent etching of (100) silicon. *Sensors and Actuators A-Physical*, 39:261–262, 1993.

[30] G. Ensell. Alignment of mask patterns to crystal orientation. *Sensors and Actuators A-Physical*, 53:345–348, 1996.

[31] Larry J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombinition. *Foundation of Genetic Algorithms*, pages 265–283, July 1991.

[32] Larry J. Eshelman, R. Caruna, and J. David Schaffer. Biases in the crossover landscape. In *The Third International Conference on Genetic Algorithms*, pages 10–19, Hillsdale, NJ, 1989. Lawrence Erlbaum.

[33] Larry J. Eshelman and J. David Schaffer. Real-coded genetic algorithms and interval-schemata. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms 1*, pages 187–202, San Mateo, CA, July 1991. International Society for Genetic Algorithms, Morgan Kaufmann, Publishers. Proceedings of the First Workshop on the Foundations of Genetic Algorithms (FOGA).

[34] Hsiao-Lan Fang, David Corne, and Peter Ross. A genetic algorithm for job-shop problems with various schedule quality criteria. In T. C. Fogarty, editor, *Evolutionary Computation, AISB Workshop*, pages 39–49, Berlin, Germany, 1996. Springer-Verlag.

[35] Gary K. Fedder. Structured design of integrated MEMS. In *12th Annual IEEE International Micro Electro Mechanical Systems Conference*, pages 1–8, Orlando, FL, January 1999. IEEE.

[36] J. M. Fitzpatrick and J. J. Grefenstette. Genetic algorithms in noisy environments. *Machine Learning*, 3:101–120, 1988.

[37] David B. Fogel. An introduction to simulated evolutionary optimization. *IEEE Transaction on Neural Networks*, 5(1):3–14, January 1994.

[38] L. J. Fogel. Antonomous automata. *Industrial Research*, 4:14–19, 1962.

[39] L. J. Fogel. *On the Organization of Intellect*. Ph.D. thesis, University of California, Los Angeles, 1964.

[40] B. Forouraghi. A genetic algorithm for multiobjective robust design. *Applied Intelligence*, 12:151–161, 2000.

[41] B. Fox and M. McMahon. Genetic operators for sequencing problems. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms 1*, pages 284–009, San Mateo, CA, July 1991. International Society for Genetic Algorithms, Morgan Kaufmann, Publishers. Proceedings of the First Workshop on the Foundations of Genetic Algorithms (FOGA).

[42] F. C. Frank and M. B. Ives. Orientation-dependent dissolution of Germanium. *Journal of Applied Physics*, 31(11):1996–1999, November 1960.

[43] J. Fruhauf and B. Hannemann. Anisotropic multi-step etch processes of silicon. *Journal of Micromechanics and Microengineering*, 7:137–140, 1997.

[44] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms And Engineering Design*. John Wiley & Sons, New York, 1996.

[45] Mitsuo Gen, K. Ida, and Runwei Cheng. Multirow machine layout problem in fuzzy environment using genetic algorithms. *Computers and Industrial Engineering*, 29:519–523, September 1995.

[46] Mitsuo Gen, B. Liu, and K. Ida. Evolution program for constrained nonlinear optimization. In *Proceedings of the 16th International Conference on Computers and Industrial Engineering*, 1994.

[47] B. Gogoi, R. Yeun, and C. H. Mastrangelo. The automatic synthesis of planar fabrication process flows for surface micromachined devices. In *Proceedings of the IEEE Micro Electro Mechanical Systems Workshop*, pages 153–157, New York, January 1994. IEEE.

[48] David E. Goldberg. Simple genetic algorithms and the minimal deceptive problem. In *Genetic Algorithms and Simulated Annealing*, pages 74–88. Morgan Kaufmann, Publishers, San Mateo, CA, 1987.

[49] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[50] David E. Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. Technical Report IlliGAL Report 90001, University of Illinios at Urbana-Champaign, Urbana, IL, 1990.

[51] David E. Goldberg and Kalyanmoy Deb. A comparative analysis of selection schemes used in genetic algorithms. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms 1*, pages 69–93, San Mateo, CA, July 1991. International Society for Genetic Algorithms, Morgan Kaufmann, Publishers. Proceedings of the First Workshop on the Foundations of Genetic Algorithms (FOGA).

[52] M. Hasanuzzaman and C. H. Mastrangelo. Process compilation of thin film microdevices. *IEEE Transaction on Computer-Aided Design of Intergrated Circuit and Systems*, 15:745–764, July 1996.

[53] John H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, 1975.

[54] Ted J. Hubbard and Erik K. Antonsson. Emergent Faces in Crystal Etching. *Journal of Microelectomechanical Systems*, 3(1):19–28, March 1994.

[55] Ted J. Hubbard and Erik K. Antonsson. Design of MEMS via Efficient Simulation of Fabrication. In *Design for Manufacturing Conference*, New York, August 1996. ASME.

[56] Ted J. Hubbard and Erik K. Antonsson. Cellular Automata in MEMS Design. *Sensors and Materials*, 9(7):437–448, 1997.

[57] Merrill Hunt and James A. Rowson. Blocking in a system on a chip. *IEEE Spectrum*, 33(11):35–41, November 1996.

[58] Phil Husbands, Giles Jermy, Malcolm McIlhagga, and Robert Ives. Two applications of genetic algorithms to component design. In T. C. Fogarty, editor, *Evolutionary Computation, AISB Workshop*, pages 50–61, Berlin, Germany, 1996. Springer-Verlag.

[59] R. J. Jaccodine. Use of modified free energy theorems to predict equilibrium growing and etching shapes. *Journal of Applied Physics*, 33(8):2643–2647, August 1962.

[60] M.J. Jakiela, C. Chapman, J. Duda, A. Adewuya, and K. Saitou. Continuum structural topology design with genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):339–356, 2000.

[61] Cezary Z. Janilow and Zbigniew Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In *The Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991. Morgan Kaufmann, Publishers.

[62] Liang kai Huang and Mao jiun J. Wang. Efficient shape matching through model-based shape recognition. *Pattern Recognition*, 29:207–215, 1996.

[63] I. Kang, M. R. Haskard, and N. D. Samaan. A study of two-step silicon anisotropic etching for a polygon-shaped microstructure using koh solution. *Sensors and Actuators A-Physical*, 62:646–651, 1997.

[64] G. Koppleman. Oyster, a simulation tool for micro electromechanical design. *Sensors and Actuators A-Physical*, 20:179–185, 1989.

[65] J. Koza. Evolution and co-evolution of computer programs to control independently-acting agents. In Jean-Arcady Meyer and Stewart W. Wilson, editors, *The First International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pages 366–375, Cambridge, MA, 1991. MIT Press.

[66] J. M. Lai, W. H. Chieng, and Y. C. Huang. Precision alignment of mask etching with respect to crystal orientation. *Journal of Micromechanics and Microengineering*, 8:327–329, 1998.

[67] Cin-Young Lee and Erik K. Antonsson. Surface Reconstruction of Etched Contours. In *MSM'99, Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators*, One Kendall Square, Cambridge, MA, U.S.A., April 1999. Applied Computational Research Society. International Conference on Modeling and Simulation of Microsystems.

[68] Hui Li. *Evolutionary Techniques Applied to Mask-layout Synthesis in Micro-Mechanical-Electronic Systems (MEMS)*. Ph.D. thesis, California Institute of Technology, Pasadena, CA, June 1999.

[69] Jens Lienig. A parallel genetic algorithm for performance-driven VLSI routing. *IEEE Transactions on Evolutionary Computation*, 1(1):29–39, April 1997.

[70] Jens Lienig. Physical design of VLSI circuits and the application of genetic algorithms. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 277–292. Springer-Verlag, Berlin, Germany, 1997.

[71] Mark K. Long, Joel W. Burdick, and Erik K. Antonsson. Design of Compensation Structures for Anisotropic Etching. In *MSM'99, Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators*, One Kendall Square, Cambridge, MA, U.S.A., April 1999. Applied Computational Research Society. International Conference on Modeling and Simulation of Microsystems.

[72] Lin Ma and Erik K. Antonsson. Mask-Layout and Process Synthesis for MEMS. In *MSM'2001, Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators*, One Kendall Square, Cambridge, MA, U.S.A., April 2000. Applied Computational Research Society. International Conference on Modeling and Simulation of Microsystems.

[73] Marc Madou. *Fundamentals of Microfabrication*. CRC Press, New York, 1997.

[74] F. Maseeh, R. Harris, and Stephen Senturia. A CAD architecture for MEMS. In *Transducers '90*, pages 44–49, New York, 1990. IEEE.

[75] Fariborz Maseeh. Intellicad MEMS computer-aided design. In Jan G. Korvink, editor, *1997 CAD for MEMS Workshop Digest*, page 18, Switzerland, March 1997. Physical Electronics Laboratory, ETH Zurich.

[76] G. K. Mayer, H. L. Offereins, H. Sandmaier, and K. Kuhl. Fabrication of non-underetched convex corners in anisotropic etching of (100)-silicon in aqueous KOH with respect to novel micromechanic elements. *Journal of the Electrochemical Society*, 137(12):3947–3951, December 1990.

[77] Pinaki Mazumder and Elizabeth M. Rudnick. *Genetic Algorithms for VLSI Design, Layout & Test Automation*. Prentice-Hall, Englewood Cliffs, NJ, 1999.

[78] C. Mead and L. Conway. *Introduction to VLSI Systems*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1980.

[79] Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Program*. Springer-Verlag, Berlin, 1994.

[80] Zbigniew Michalewicz, T. Logan, and S. Swaminathan. Evolutionary operations for continuous convex parameter spaces. In Anthony V. Sebald and Lawrence J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 84–97, Singapore, 1994.

[81] B. L. Miller and D. E. Goldberg. Genetic algorithms, selection scheme, and the varying effect of noise. *Evolutionary Comutation*, 4:113–131, 1996.

[82] Tamal Mukherjee, Gary K. Fedder, and R. D. (Shawn) Blanton. Hierarchical design and test of integrated microsystems. *IEEE Design and Test of Computers*, 16(4):18–27, Oct.-Dec. 1999.

[83] D. Mumford. The problem of robust shape descriptors. In *Proceedings of First International Conference on Computer Vision*, 1987.

[84] S. Obayashi, D. Sasaki, Y. Takeguchi, and N. Hirose. Multiobjective evolutionary computation for supersonic wing-shape optimization. *IEEE Transactions on Evolutionary Computation*, 4(2):182–187, 2000.

[85] W. G. Oldham, S. N. Nandgaonkar, A. R. Neureurather, and M. M. O'Toole. A general simulator for vlsi lithography and etching process: Part i - application to projection lithography. *IEEE Transactions on Electron Devices*, 26:717–722, 1979.

[86] W. G. Oldham, A. R. Neureurather, C. Sung, J. L. Reynolds, S. N. Nandgaonkar, and M. M. O'Toole. A general simulator for vlsi lithography and etching process: Part ii - application to deposition and etching. *IEEE Transaction on Electron Devices*, 27:1455–1559, 1980.

[87] Kevin N. Otto and Erik K. Antonsson. Trade-Off Strategies in Engineering Design. *Research in Engineering Design*, 3(2):87–104, 1991.

[88] Kevin N. Otto and Erik K. Antonsson. Extensions to the Taguchi Method of Product Design. *ASME Journal of Mechanical Design*, 115(1):5–13, March 1993.

[89] D. B. Parkinson. Simulated variance optimization for robust design. *Quality and Reliability Engineering International*, 14:15–21, 1998.

[90] T. P. Pavlidis. Polygonal approximation by Newton's method. *IEEE Transaction Computation*, vol. C-26 No. 8:800–807, 1977.

[91] Andy Perrin, Venkat Ananthakrishnan, Feng Gao, Radha Sarma, and G. K. Ananthasuresh. Voxel-based heterogeneous geometric modeling for surface micromachined MEMS. In *MSM'2001, Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators*, One Kendall Square, Cambridge, MA, U.S.A., March 2001. Applied Computational Research Society. International Conference on Modeling and Simulation of Microsystems.

[92] M. Phadke. *Quality Engineering Using Robust Design*. Prentice Hall, Englewood Cliffs, NJ, 1989.

[93] F. Pourahmadi and J. Twerdok. Modeling micromachined sensors with finite elements. *Machine Design*, pages 44–60, July 1990.

[94] B. Puers and W. Sansen. Compensation structures for convex corner micromachining in silicon. *Sensors and Actuators A-Physical*, 21-23:1036–1041, 1990.

[95] N. Radcliffe. *Genetic Neural Networks on MIMD Computers*. Ph.D. thesis, University of Edinburgh, UK, 1990.

[96] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biolgischen evolution*. Stuttgart: Frommann-Holzboog Verlag, 1973.

[97] Peter Ross and Dave Corne. Comparing genetic algorithms, simulated annealing, and stochastic hill-climbing on timetable problems. In T. C. Fogarty, editor, *Evolutionary Computation, AISB Workshop*, pages 94–102, Berlin, Germany, 1995. Springer-Verlag.

[98] Kazuo Sato, Mitsuhiro Shikida, Yoshihiro Matsushima, Takishi Yamashiro, Kazuo Asaumi, Yasuroh Iriye, and Masaharu Yamamoto. Characterization of orientation-dependent etching properties of single-crystal silicon: Effects of KOH concentration. *Sensors and Actuators A-Physical*, 64:87–93, 1998.

[99] Dragan A. Savic and Godfrey A. Walters. Genetic operators and constraint handling for pipe network optimization. In T. C. Fogarty, editor, *Evolutionary Computation, AISB Workshop*, pages 154–165, Berlin, Germany, 1995. Springer-Verlag.

[100] H. P. Schwefel. *Kybernetische evolution als strtegie der experimentellen forschung in der strmungstechnik*. Ph.D. thesis, Technical University of Berlin, 1965.

[101] Michael J. Scott and Erik K. Antonsson. Aggregation Functions for Engineering Design Trade-offs. *Fuzzy Sets and Systems*, 99(3):253–264, 1998.

[102] Michael J. Scott and Erik K. Antonsson. Using Indifference Points in Engineering Decisions. In *11th International Conference on Design Theory and Methodology*. ASME, September 2000.

[103] H. Seidel, L. Csepregi, A. Heuberger, and H. Baumgartel. Anisotropic etching of crystaline silicon in alkaline solutions. *Journal of the Electrochemical Society*, 137:3613–3631, 1990.

[104] Stephen D. Senturia, N. Aluru, and Jacob K. White. Simulating the bahavior of mems devices: Computational methods and needs. *IEEE Computational Science and Engineering*, 4:30–43, January 1997.

[105] Stephen D. Senturia, R. M. Harris, B. P. Johnson, S. Kim, M. A. Shulman, and Jacob K. White. A computer-aided design system for microelectromechanical systems (MEMCAD). *Journal of Microelectomechanical Systems*, 1:3–13, March 1992.

[106] C. H. Sequin. Computer simulation of anisotropic crystal etching. *Sensors and Actuators A-Physical*, 34(3):225–241, September 1992.

[107] D. W. Shaw. Morphology analysis in localized crystal growth and dissolution. *Journal of Crystal Growth*, 47:509–517, 1979.

[108] D. W. Shaw. Localized etching with acidic hyrogen peroxide solutions. *Journal of the Electrochemical Society: Solid State Science and Technology*, pages 874–880, April 1981.

[109] Mitsuhiro Shikida, Kazuo Sato, Kenji Tokoro, and Daisuke Uchikawa. Comparison of anisotropic etching properties between KOH and TMAH solutions. *Journal of Microelectomechanical Systems*, 1999.

[110] William M. Spears and Kenneth De Jong. On the virtues of parameterized uniform crossover. In *The Fourth International Conference on Genetic Algorithms*, pages 230–236, San Mateo, CA, 1991.

[111] Gilbert Syswerda. Uniform crossover in genetic algorithms. In *The Fourth International Conference on Genetic Algorithms*, pages 2–9, San Mateo, CA, 1989.

[112] G. Taguchi. *Introduction to Quality Engineering*. Asian Productivity Organization, Unipub, White Plains, NY, 1986.

[113] William C. Tang. Overview of microelectromechanical systems and design processes. In *Proceedings of the 34th Design Automation Conference*, pages 670–673, New York, June 1997. SIGDA, Association for Computing Machinery (ACM). Paper 42.1.

[114] O. Than and S. Buttgenbach. Simulation of anisotropic chemical etching of crytalline silicon using a cellular-automata model. *Sensors and Actuators A-Physical*, 45:85–89, 1994.

[115] T. Thurgate. Segment based etch algorithm and modeling. *IEEE Transactions on Computer-Aided Design*, 10(9):1101–1109, September 1991.

[116] S. Tsutsui and A. Ghosh. Genetic algorithms with a robust solution searching scheme. *IEEE Transactions on Evolutionary Comutation*, 1:201–208, 1997.

[117] Mattias Vangbo and Ylva Backlund. Precise mask alignment to the crystallographic orientation of silicon wafers using wet anisotropic etching. *Journal of Micromechanics and Microengineering*, 6:279–284, 1996.

[118] Kensall D. Wise. Integrated microelectromechanical systems: A perspective on MEMS in the 90s. In *MEMS '91*, pages 33–38, New York, 1991. IEEE.

[119] Alden H. Wright. Genetic algorithms for real parameter optimization. In Gregory J. E. Rawlins, editor, *Foundations of Genetic Algorithms 1*, pages 205–217, San Mateo, CA, July 1991. International Society for Genetic Algorithms, Morgan Kaufmann, Publishers. Proceedings of the First Workshop on the Foundations of Genetic Algorithms (FOGA).

[120] X. Wu and W. H. Ko. Compensating corner undercutting in anisotropic etching of (100) silicon. *Sensors and Actuators A-Physical*, 18:207–215, 1989.

[121] Heng Yang, Minhang Bao, Shaoqun Shen, Xinxin Li, Dacheng Zhang, and Guoyin Wu. A novel technique for measuring etch rate distribution of Si. *Sensors and Actuators A-Physical*, 79:136–140, 2000.

[122] Kelvin K. Yue and David J. Lilja. Designing multiprocessor scheduling algorithms using a distributed genetic algorithm system. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 207–222. Springer-Verlag, Berlin, Germany, 1997.

[123] QingXin Zhang, Litian Liu, and Zhijian Li. A new approach to convex corner compensation for anisotropic etching of (100) Si in KOH. *Sensors and Actuators A-Physical*, 56:251–254, 1996.

[124] Y. Zhang, Selden B. Crary, and Kensall D. Wise. Pressure sensor design and simulation using the CAEMEMS-D module. In *Solid-State Sensor and Actuator Workshop*, pages 32–35, New York, 1990. Transducers Research Foundation, Inc., IEEE. Technical Digest.

[125] Zhenjun Zhu and Chang Liu. Anisotropic crystalline etching simulation using a continuous cellular automata algorithm. In *Micro-Electro-Mechanical Systems (MEMS)*, pages 577–582, New York, NY, November 1998. ASME. International Mechanical Engineering Congress and Exposition (IMECE), Anaheim, CA, DSC-Vol. 66.

[126] Zhenjun Zhu and Chang Liu. Micromachining process simulation using a continuous cellular automata method. *Journal of Microelectomechanical Systems*, 9(2):252–261, June 2000.